(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

COMPUTATIONAL INVESTIGATION OF THE COMPRESSIBLE DYNAMIC STALL CHARACTERISTICS OF THE SIKORSKY SSC-A09 AIRFOIL

by

Thomas A. Johnston

September, 1993

Thesis Advisor:	M. F. Platzer
Co-Advisor:	J. A. Ekaterinaris

Approved for public release; distribution is unlimited.

Unclassified
Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification: Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution/Availability of Report Approved for public release; distribution is unlimited.
2b Declassification/Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School — 6b Office Symbol *(if applicable)* AA	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey CA 93943-5000	7b Address *(city, state, and ZIP code)* Monterey CA 93943-5000
8a Name of Funding/Sponsoring Organization — 6b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number
Address *(city, state, and ZIP code)*	10 Source of Funding Numbers Program Element No / Project No / Task No / Work Unit Accession No

11 Title *(include security classification)* COMPUTATIONAL INVESTIGATION OF THE COMPRESSIBLE DYNAMIC STALL CHARACTERISTICS OF THE SIKORSKY SSC-A09 AIRFOIL

12 Personal Author(s) Johnston, Thomas A.

13a Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
Master's Thesis	From To	September 1993	261

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	Dynamic Stall, Supercritical Airfoil, Navier-Stokes Solution, Shock-induced Boundary Layer Separation

19 Abstract *(continue on reverse if necessary and identify by block number)*

Steady and unsteady two-dimensional flowfield analysis was conducted for a Sikorsky SSC-A09 airfoil in compressible, high Reynolds number flows. Limited verification with experimental measurement was achieved. Computational methods included a steady, linear panel method with compressibility corrections; a laminar and turbulent boundary layer method; an unsteady, linear panel method; and a numerical solution method of the thin layer, compressible, Navier-Stokes equations using a body-fitted C-type computational grid. The Baldwin-Lomax, two-layer, zero-equation turbulence model was used. Wind tunnel wall interference effects were ignored. Steady and unsteady airloads and instantaneous flow pictures are presented. In steady flow with little or no separation, computed lift, drag, pitching moment, and skin friction coefficients, as well as displacement thickness and boundary layer velocity profiles at several angles-of-attack were generally found to be in good agreement with experimental data.

20 Distribution/Availability of Abstract X unclassified/unlimited — same as report — DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Platzer, M. F.	22b Telephone *(include Area Code)* 408-656-2058	22c Office Symbol AA/Pl

DD FORM 1473,84 MAR — 83 APR edition may be used until exhausted — security classification of this page
All other editions are obsolete — Unclassified

COMPUTATIONAL INVESTIGATION OF THE COMPRESSIBLE DYNAMIC STALL CHARACTERISTICS OF THE SIKORSKY SSC-A09 AIRFOIL

by

Thomas A. Johnston
Commander, United States Navy
B.S., United States Naval Academy, 1978

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 1993

Author: Thomas A. Johnston

Thomas A. Johnston

Approved by: M. F. Platzer

M. F. Platzer, Thesis Advisor

J. A. Ekaterinaris, Co-Advisor

D. J. Collins

D. J. Collins, Chairman
Department of Aeronautics and Astronautics

ABSTRACT

Steady and unsteady two-dimensional flowfield analysis was conducted for a Sikorsky SSC-A09 airfoil in compressible, high Reynolds number flows. Limited verification with experimental measurement was achieved. Computational methods included a steady, linear panel method with compressibility corrections; a laminar and turbulent boundary layer method; an unsteady, linear panel method; and a numerical solution method of the thin layer, compressible, Navier-Stokes equations using a body-fitted C-type computational grid. The Baldwin-Lomax, two-layer, zero-equation turbulence model was used. Wind tunnel wall interference effects were ignored. Steady and unsteady airloads and instantaneous flow pictures are presented. In steady flow with little or no separation, computed lift, drag, pitching moment, and skin friction coefficients, as well as displacement thickness and boundary layer velocity profiles at several angles-of-attack were generally found to be in good agreement with experimental data.

TABLE OF CONTENTS

TABLE OF SYMBOLS

a	Speed of Sound
A	Reduced Pitch Rate; = $(\dot{\alpha}c/2U)$
c	Airfoil Chord length
C_d	Section Pressure Drag Coefficient
C_F	Skin Friction Coefficient
C_l	Section Lift Coefficient
C_m	Section Moment Coefficient
C_P	Pressure Coefficient; = $(P-P_\infty)/q$
C_x	X-Force Coefficient
C_y	Y-Force Coefficient
e	Total Energy per Unit Volume
F	Inviscid Flux Vector, ξ
G	Inviscid Flux Vector, ς
k	Reduced Frequency; = $(\omega c/2U)$
M	Mach Number
$\hat{n}$	Unit Normal Vector
N	Total Number of Panels
p	Pressure
P_r	Prandtl Number
q	Dynamic Pressure; = $\frac{1}{2}\rho U^2$
q(s)	Source Strengths
Q	Conservative Variables Vector
r	Scalar Distance between two points
R_e	Reynolds Number
S	Viscous Fluxes
$\hat{t}$	Unit Tangential Vector
u	Velocity Component, x-direction
U	Freestream Velocity Magnitude
v	Velocity Component, y-direction
α	Geometric Angle-of-Attack
α_{L-0}	Angle-of-Attack at Zero Lift

γ	Ratio of Specific Heats
$\gamma(s)$	Vortex Strengths
γ_{tr}	Intermittency Factor
Γ	Circulation
δ^*	Displacement Thickness
ϵ_m	Turbulent Eddy-Viscosity
μ	Dynamic Viscosity
ν	Kinematic Viscosity
ρ	Density
ϕ	Velocity Potential
ω	Oscillation Frequency

Subscript Indices:

i,j	Indicator for Airfoil Panels and Nodes
n	Normal Component
k	Time indice
t	Tangential Component
tr	Transition
x	X-Component
y	Y-Component
∞	Free Stream Static

Operators:

∂	Partial Derivative
∇	Gradient
∇^2	Laplacian
Σ	Summation
$\int$	Integral
$\oint$	Surface Integral
U_{xx}	Second Derivative with Respect to X
U_{yy}	Second Derivative with Respect to Y

I. INTRODUCTION

Historically aeronautical engineers have had only wind tunnel and flight test experiments to validate aerodynamic theory, often at great expense. In today's world of powerful supercomputers and advanced personal computers with vast memory capability, flowfield solutions once thought impossible or prohibitively expensive are becoming feasible in this new age of Computational Fluid Dynamics (CFD). CFD has become an effective research tool in understanding complicated fluid dynamics phenomena. Indeed, as illustrated in Figure 1.1, the Theory, Experiment, and CFD triad complement each other as

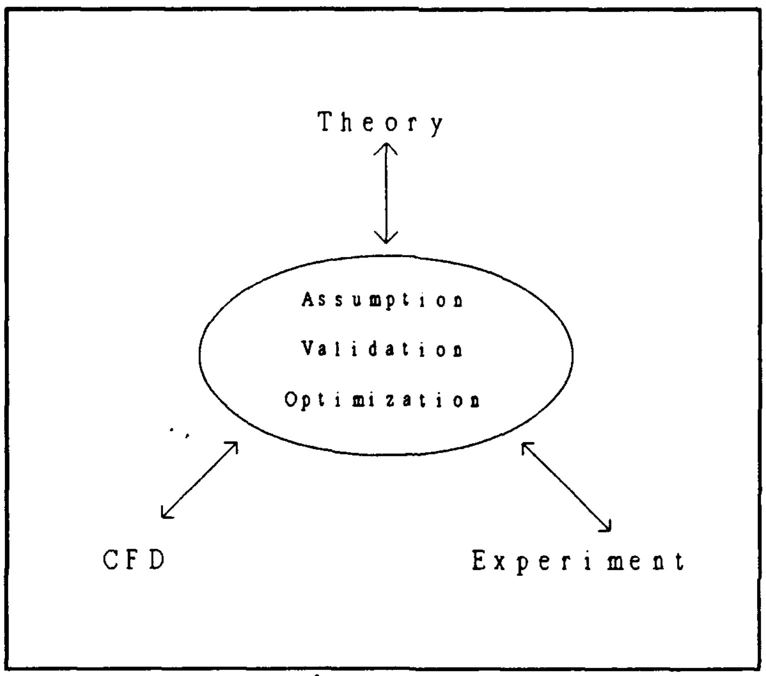


Figure 1.1

well as add a new dimension to the validation process. Experiment and CFD can now be used to complement and optimize each other. CFD results allow in-depth understanding of many physical processes. Transition and turbulence models can be verified and code robustness (convergence time span) can be optimized. Within the CFD framework, it was once thought that the full Navier-Stokes equations would have to be solved to obtain realistic flows over an airfoil executing maneuvers in a viscous, compressible medium. Numerical scheme accuracy and convergence rates are complicated by the various length scales of the viscous effects near the airfoil and those of the surrounding inviscid flow field. It has been proven that more cost effective solution methods can be employed that make realistic simplifications to the governing equations and allow moving away from the supercomputer to the personal computer thus yielding beneficial results at greatly reduced time and cost. Ultimately, the overall goal would be the completion of the design process using only CFD methods.

A current field of intense investigation is the aerodynamics of a rapidly pitching airfoil. Two effects are of major interest:

- Augmented lift created during dynamic stall while performing aircraft combat maneuvers (ACM) in high Reynolds number flows.
- Dynamic stall on a retreating helicopter rotor blade during high-speed forward flight.

Dynamic lift and stall are dominated by the generation of a vortex near the leading edge of the suction surface and its subsequent convection over the airfoil surface. This sequence of events is discussed and shown in great detail in Chapter V.

The intent of this thesis is the CFD investigation of the unsteady aerodynamics of a Sikorsky SSC-A09 airfoil undergoing high pitch rate maneuvers. The experimental results of Lorber and Carta [Ref. 11] are used for validation of the computed solutions. The investigation goals are:

- Determine the influence of the leading edge stall vortex on the unsteady aerodynamic response during and after stall.
- Determine the location of any separation bubbles.
- Determine the location and extent of the boundary layer transition.
- Determine compressibility effects in inviscid and viscous flows.
- Determine the effect of any supersonic regions and shock waves created during pitch up ramp or sinusoidal maneuvers.
- Accurately predict pressure loads, forces, and moments.
- Determine the most efficient and cost effective CFD approach that achieves the desired level of accuracy.

In the following sections the methods which were used to analyze the above flow phenomena are presented first. A presentation of the numerical results and comparisons with experiment follows. Each section includes an Appendix which contains a complete user's guide for the reader who wishes to apply the codes to similar problems. Finally, a discussion of all the results is presented and some conclusions with recommendations for future research are given.

II. STEADY, LINEAR PANEL CODE

A. POTENTIAL FLOW THEORY/BACKGROUND

The flow field is assumed to be steady, incompressible, inviscid and irrotational. A steady flow field implies the fluid velocity and pressure depend only on the spatial coordinates and not on time. Flow field incompressibility implies that the divergence (the time rate of change of volume of a moving fluid element per unit volume) of the velocity vector is zero as indicated in Equation 2.1, and that density is a constant throughout.

$$\nabla \bullet \vec{V} = 0 \tag{2.1}$$

Flow field irrotationality implies that vorticity is zero everywhere, Equation 2.2, and that a scalar function must exist such that the velocity is given by the scalar function's gradient as shown in Equation 2.3.

$$\nabla X \vec{V} = 0 \tag{2.2}$$

$$\nabla \phi = \vec{V} \tag{2.3}$$

Consequently, irrotational flows are often described as 'potential flows'.

A flow field that is both incompressible and irrotational must satisfy Laplace's equation:

$$\nabla^2\phi = \phi_{xx} + \phi_{yy} = 0 \tag{2.4}$$

Since Laplace's equation is a linear homogeneous second order partial differential equation, the principle of superposition holds. Complicated flows can be created by linearly combining elementary flows that are both incompressible and irrotational. Uniform, source, and vortex flows are examples that meet these conditions (Anderson [Ref.2]).

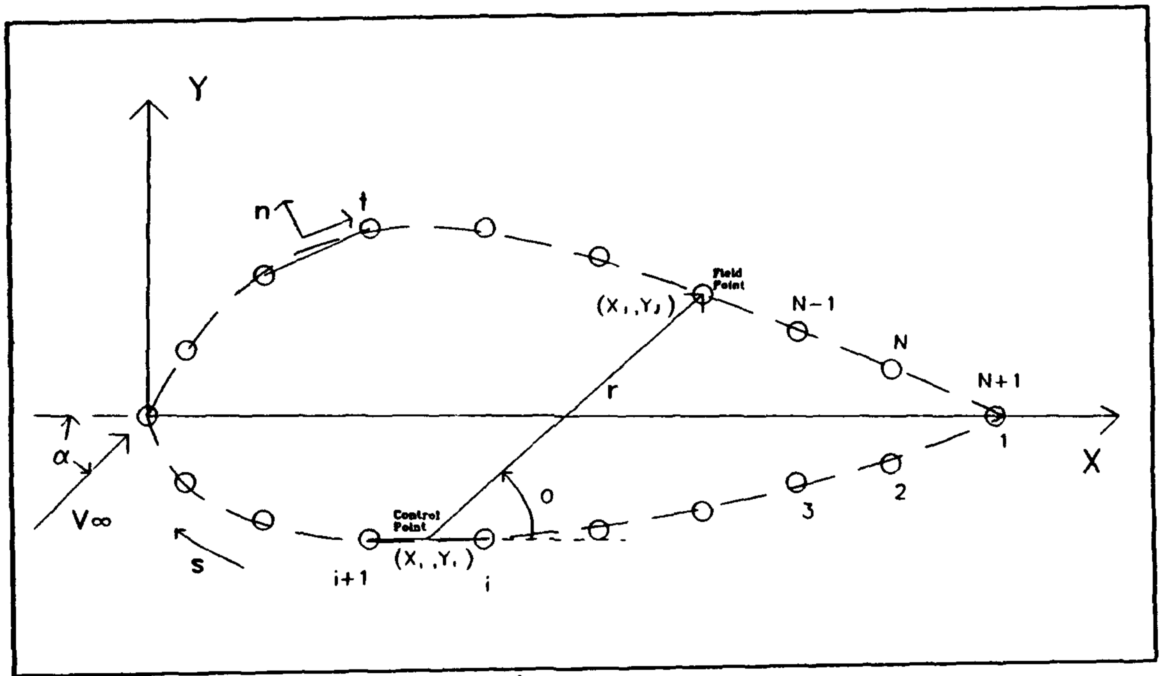


Figure 2.1
Airfoil Geometry and Coordinate System

1. Reference Frame

The two-dimensional airfoil geometry and (x,y) and (r,θ) coordinate systems are described in Figure 2.1. The airfoil surface is divided into a number (N) of straight line

segments normally called 'panels'. N+1 surface points, normally called nodes, distinguish the N panels. Numbering convention starts from the lower trailing edge and proceeds clockwise around the airfoil making the first and last point the same. Panel length is arbitrary, but enforcement of the trailing edge Kutta condition (the trailing edge flow must depart smoothly since it is a stagnation point) requires that the first and last panel length be the same. Unit normal vectors, $\hat{n}$, are perpendicular, positive outward from the panel surface. Unit tangent vectors, $\hat{t}$, are parallel to the panel surface, positive in the clockwise direction.

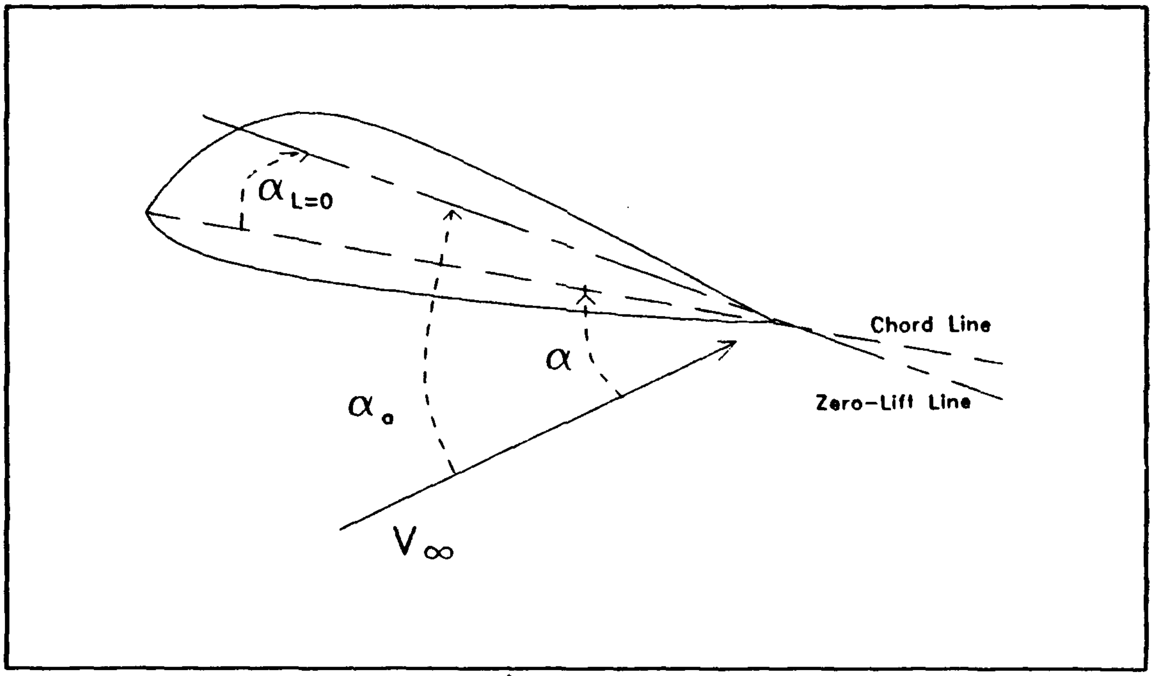


Figure 2.2
Angle-of-Attack Standardization

2. Airfoil Nomenclature

Standard airfoil nomenclature is used as displayed in Figure 2.2. Summary of nomenclature (Kuethe and Chow [Ref. 10]):

- **Chord Line.** The straight line connecting the leading and trailing edges.
- **Chord.** The distance between the leading and trailing edges along the chord line.
- **Mean Camber Line.** The locus of points one-half way between the upper and lower surface measured perpendicular to the mean camber line itself.
- **Symmetric Airfoil.** An airfoil where the mean camber and chord lines are the same.
- **Aerodynamic Center.** The point on the airfoil where the moment is independent of angle-of-attack.
- **Center of Pressure.** The location where the resultant of a distributed load effectively acts on a body. The point about which the aerodynamic moment is zero.
- **Geometric Angle-of-Attack (α).** The angle between V_∞ and the chord line.
- **Zero-Lift Line.** A line on the airfoil parallel to the flight path and passing through the trailing edge when the airfoil is oriented to create zero lift. The zero-lift and chord line are the same for a symmetric airfoil.

- **Angle-of-Attack at Zero Lift ($\alpha_{L=0}$).** The angle between the chord and zero-lift lines.
- **Absolute Angle-of-Attack (α_a).** The angle between V_∞ and the zero-lift line.

$$\alpha_a = \alpha - \alpha_{L=0} \tag{2.5}$$

3. **Singularity Distribution**

The airfoil velocity potential (Φ) is determined by decomposing the potential flow field into a free stream flow, and placing a source and vortex distribution at each control point (mid point) of each panel. Vortex flows provide circulation/lift and here vortex strength (γ) is fixed. Source flows accurately represent body thickness. Source distributions (q) are allowed to vary from panel to panel. The total potential is described below. These integrals are calculated along the surface contour s in polar coordinates.

$$\Phi_{total} = \phi_\infty + \phi_{source} + \phi_{vortex} \tag{2.6}$$

$$\phi_\infty = V_\infty \times [\, x \cos\alpha + y \sin\alpha \,] \tag{2.7}$$

$$\phi_{source} = \int_s \left\{ \frac{q(s)}{2\pi} \ln r \right\} ds \tag{2.8}$$

$$\phi_{vortex} = - \int_s \left\{ \frac{\gamma(s)}{2\pi} \, \theta \right\} ds \tag{2.9}$$

Integration is performed on each panel along a straight line where q_i and γ are constant and then all the panels summed. The velocity is then obtained from $\nabla\Phi$.

4. Influence Coefficients

Influence coefficients provide an algebraic system of linear simultaneous equations that ease numerical solution. An influence coefficient is defined by the velocity induced at a field point (on the airfoil surface) by a unit strength singularity (Source and Vortex) distribution on one panel. Nowak [Ref. 13], Teng [Ref. 15], and Tuncer [Ref. 16] provide detailed analysis of geometrical quantities, equations and the numerical solution scheme.

a. Boundary Conditions

Two boundary conditions must be satisfied. The first is the flow tangency condition at all control points (the mid point of each panel). This is accomplished by requiring the normal component of velocity at the control point to be zero for all panels. The second, the Kutta condition requires smooth flow leaving the trailing edge, and is accomplished by equating the upper and lower pressures at the trailing edge. This is enforced by equating the tangential velocities on the first and N^{th} panel.

5. Coefficient of Pressure (C_p)

Once the source strengths (q_i) and vortex strength (γ) are calculated, the normalized velocity $(V_{total}/V_\infty)_i$ is computed

at each control point. Using Bernoulli's equation, Equation 2.10, the incompressible flow Coefficient of Pressure is computed.

$$C_p = \frac{P - P_\infty}{q_\infty} = 1 - \left\{\frac{V_{total}}{V_\infty}\right\}^2 \qquad \textbf{(2.10)}$$

a. Pressure Compressibility Correction

For low Mach number flows, less than M=.3, the density variation in an inviscid flow is negligible (less than a 5% variation, Anderson [Ref. 2]). For higher, subsonic (M_∞<.7) Mach number flows, a compressibility correction to the incompressible data is achieved by using the 'Prandtl-Glauert' rule derived from small perturbation, linearized velocity potential theory:

$$C_{P_{comp}} = \frac{C_{P_{incomp}}}{\sqrt{1 - M_\infty^2}} \qquad \textbf{(2.11)}$$

6. Force and Moment Coefficients

The force and moment coefficients are computed by integration/summation of the pressure distribution assuming a constant C_P on each panel. The total force on a single panel would be $\mathbf{C_{Pi} \bullet ds}$. Figure 2.3 details the required geometry. Airfoil-fixed forces for panel **i** are:

$$\sin\beta = \frac{dy}{ds} \qquad \cos\beta = \frac{dx}{ds} \tag{2.12}$$

$$C_{F_{x_i}} = C_{P_i} \times ds \times \sin\beta = C_{P_i} \times dy \tag{2.13}$$

$$C_{F_{Y_i}} = C_{P_i} \times ds \times \cos\beta = C_{P_i} \times dx \tag{2.14}$$

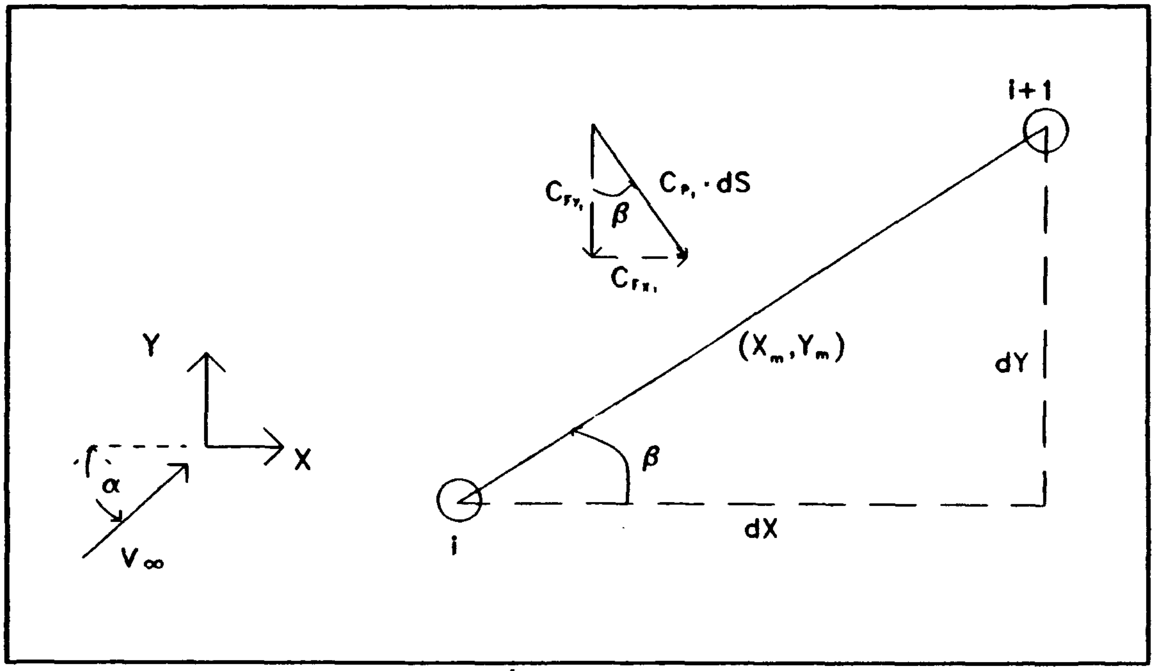


Figure 2.3
Force and Moment Geometry

Integration of forces over N panels with respect to the airfoil-fixed coordinate system (negative sign on C_{Fx} due to the sign convention of positive tangential velocities in the clockwise direction around the airfoil) are shown in Equations 2.14 and 2.15:

$$C_{F_x} = - \sum_{i=1}^{N} C_{P_i} \{ y_{i+1} - y_i \} \qquad (2.15)$$

$$C_{F_y} = \sum_{i=1}^{N} C_{P_i} \{ x_{i+1} - x_i \} \qquad (2.16)$$

Rotation with respect to the free stream direction (tangent for drag and perpendicular for lift) is described in Equations 2.17 and 2.18:

$$C_l = C_{F_y} \cos\alpha - C_{F_x} \sin\alpha \qquad (2.17)$$

$$C_d = C_{F_x} \cos\alpha + C_{F_y} \sin\alpha \qquad (2.18)$$

The moment is taken about the quarter-chord point from each control point (x_m, y_m) and summed:

$$C_m = \sum_{i=1}^{N} C_{P_i} \left\{ (x_{i+1} - x_i) (x_{m_i} - .25) + (y_{i+1} - y_i) y_{m_i} \right\} \qquad (2.19)$$

7. Thin Airfoil Theory Prediction

One of the basic assumptions of two-dimensional inviscid theory holds that the flow always closes smoothly and completely around the trailing edge, therefore, integrally producing **zero pressure drag** (d'Alembert's paradox). Drag is primarily due to viscous effects which generate frictional

shear forces at the surface eventually causing flow separation. D'Alembert's paradox is also true for subsonic compressible flow since the compressible and incompressible pressure coefficients differ only by a constant. This can be proven since there is no locally supersonic flow that would create wave drag. Inviscid flow theory has proven to be in good agreement with experiment in the linear region of the C_{l_α} curve where there is no flow separation (Anderson [Ref. 2]). Thin airfoil theory predicts that for a symmetric airfoil:

- The lift-curve-slope is 2π.
- The aerodynamic center and the center of pressure are at the quarter-chord point.

For a cambered airfoil:

- The lift-curve-slope is 2π.
- Only the aerodynamic center is at the quarter-chord point and the center of pressure varies with C_l.

8. The Supercritical Airfoil

Airfoil quality and efficiency are measured by its L/D which determines aerodynamic efficiency and $C_{L_{max}}$ which determines stall speed and is critically dependent upon airfoil thickness. The supercritical airfoil was the result of Richard T. Whitcomb's (working at The National Aeronautics and Space Administration - NASA) development of two-dimensional turbulent airfoils with good transonic behavior, improved drag divergence Mach numbers, and good low-speed

maximum lift and stall characteristics. The concept was based on obtaining locally supersonic flow on the upper surface with an isentropic recompression. As the airflow approaches the speed of sound, a local area of supersonic flow extending vertically appears over the upper surface. On a conventional airfoil, this flow would terminate in a shock wave at about mid-chord producing significant losses. Separation of the boundary layer is then aggravated by the shock induced pressure rise superimposing on an adverse pressure gradient. The supercritical airfoil allows the shock to position itself significantly aft of mid-chord producing a more even upper surface pressure distribution. The resulting airfoil series was characterized by a large leading edge radius, less curvature across the upper surface middle region (limiting flow acceleration), and aft camber where its influence is a maximum (Harris [Ref. 8]).

B. CODE VALIDATION

1. Computer Codes

Many panel codes based on steady, incompressible, inviscid flow over arbitrary airfoils have been developed. This paper uses versions written, and subsequently modified, by Nowak [Ref. 13] and Teng [Ref. 15]. Required input consists of angle-of-attack in degrees and the number of airfoil panels. Normalized velocities and pressure coefficients at each control point are produced. A complete

users guide for the Airfoil.f and Panel.f programs are provided in Appendix A.

2. The NACA 0012 Symmetric Airfoil

a. Geometry and Output Verification

A NACA 0012 symmetric airfoil was chosen to investigate the effects of increasing panel number and compressibility. Figure 2.4 illustrates a typical 100 panel airfoil generated by the airfoil.f program. Excellent agreement between calculated Pressure Coefficient and results obtained by Anderson [Ref. 2] at 9° angle-of-attack is shown in Figure 2.5. Compressibility effects, an increased suction peak with increasing Mach number, on a symmetric airfoil are also demonstrated.

b. Forces and Moment Comparison

Lift, drag, and pitching-moment coefficient as a function of angle-of-attack and panel number are displayed in Figures 2.6 through 2.8. The number of panels has little effect on calculated lift and only a slight effect on calculated moment. However, Figure 2.8 graphically displays the wide variation of calculated drag as a function of panel number. It is important to note that this 'calculated' drag is not real. In reality, the integral pressure drag should be zero as indicated in section A.7. As this figure illustrates, the suction peak forces cannot be exactly resolved when a summation is made over N discrete panels. Further

investigation, Figure 2.9, reveals that airfoil thickness also plays an important role. The leading edge suction peak is more easily resolved on thicker airfoils. Compressibility effects are displayed in Figures 2.10 through 2.12 - Lift, drag, and pitching-moment coefficient magnitude increase with increasing Mach number.

c. The Aerodynamic Center

Thin airfoil theory predicts the aerodynamic center to be at the quarter-chord point (section A.7). Figure 2.13 illustrates the pitching-moment coefficient as a function of Mach number and pivot point (the point about which all moments are taken). The aerodynamic center was located at 26.05% for both M=0.2 and M=0.4. Kuethe and Chow [Ref. 10] state that the position of the aerodynamic center is a function of airfoil thickness, geometry (camber), and viscosity. Here the thickness effect is seen as moving the aerodynamic center aft.

3. The Eppler E585 Airfoil

This airfoil was designed for sailplanes in low Reynolds number flows. A 71 panel geometry is displayed in Figure 2.14. The angle-of-attack for zero lift is 5.53°. Good agreement was achieved between the panel code calculation and Eppler's [Ref. 7] measured velocity distributions, Figure 2.15. Only slight variation was identified at the trailing edge that is easily resolved by splining in additional panels. Figures 2.16 through 2.19 display compressibility effects on

this cambered airfoil. Compressibility enhances lift and a more pronounced suction peak is observed.

C. THE SIKORSKY SSC-A09 SUPERCRITICAL AIRFOIL

1. Airfoil Geometry

This is a 9% thick, supercritical airfoil (section A.8) used in the Lorber and Carta experiment [Ref. 11]. The original geometry consisted of 132 surface coordinates (131 panels) as shown in Figure 2.20. The trailing edge was modified, Figure 2.21, to meet Kutta condition requirements: A sharp trailing edge, and the first and last panel having the same length. The resulting surface coordinates were manually entered into the points.dat input file.

2. Lorber and Carta Experimental Data

Lorber and Carta [Ref. 11] completed an experiment studying the aerodynamics of dynamic stall penetration at constant pitch rate and free stream Mach numbers of 0.2 through 0.4 corresponding to a Reynolds number of two through four million using the Sikorsky SSC-A09 airfoil. The two-dimensional tunnel experiment obtained dynamic stall data at conditions representative of full-scale helicopter rotor blades and maneuverable combat aircraft. A 17.3 inch chord wing was oscillated in pitch using both ramp and sinusoid motion. Wind tunnel wall effects were not accounted for.

Detailed aerodynamic response was obtained from 72 miniature pressure transducers and eight surface hot film

gages. Unsteady data included 36 constant speed ramps and nine sinusoidal oscillations. Ramp motion was a modified motion consisting of an initial delay, a constant rate increase to maximum, and then a second delay at maximum. Force and pitching-moment coefficients were determined by integrating pressures over the airfoil using the following:

$$C_N = \frac{1}{qc} \int (P_{low} - P_{up})\, dx \tag{2.20}$$

$$C_C = \frac{1}{qc} \int (P_{low} - P_{up}) \frac{dy}{dx}\, dx \tag{2.21}$$

$$C_M = \frac{1}{qc^2} \int (P_{low} - P_{up})\,(x - 0.25c)\, dx \tag{2.22}$$

$$C_L = C_N \cos\alpha - C_C \sin\alpha \tag{2.23}$$

$$C_D = C_C \cos\alpha + C_N \sin\alpha \tag{2.24}$$

3. Panel Number Effects on Forces and Moment

Lift, drag, and pitching-moment coefficient as a function of panel number and angle-of-attack are illustrated in Figures 2.22 through 2.24. Panel density was evenly increased around the leading edge using a spline program to stimulate peak suction resolution. A total of 184 panels was found to minimize the calculated drag resulting in a maximum C_d of .004 at 15° angle-of-attack. As before, the lift

coefficient was found insensitive and the moment coefficient was found to be only slightly sensitive to panel number.

4. Force and Moment Results

Panel computed and Lorber and Carta measured pressure coefficient as a function of Mach number and angle-of-attack is illustrated in Figures 2.25 through 2.32. Reasonable agreement was achieved at small angles-of-attack (0° to 9°). Increasing angle-of-attack and using compressibility corrections caused deviation from measured values.

Lift coefficient as a function of Mach number and angle-of-attack for calculated and Lorber and Carta measured values are displayed in Figure 2.33. Only slight deviation is observed at M=0.2 through 10° angle-of-attack. However, the compressibility effect calculated by panel.f was in the opposite direction (increasing $C_{L\alpha}$ with increasing Mach number) to that measured by Lorber and Carta.

Moment coefficient as a function of Mach number, angle-of-attack, and pivot point calculated by panel.f and measured by Lorber and Carta are displayed in Figures 2.34 and 2.35. The general compressibility effect is accurately predicted and good correlation was achieved at lower angle-of-attacks. The aerodynamic center was located at 25.05%.

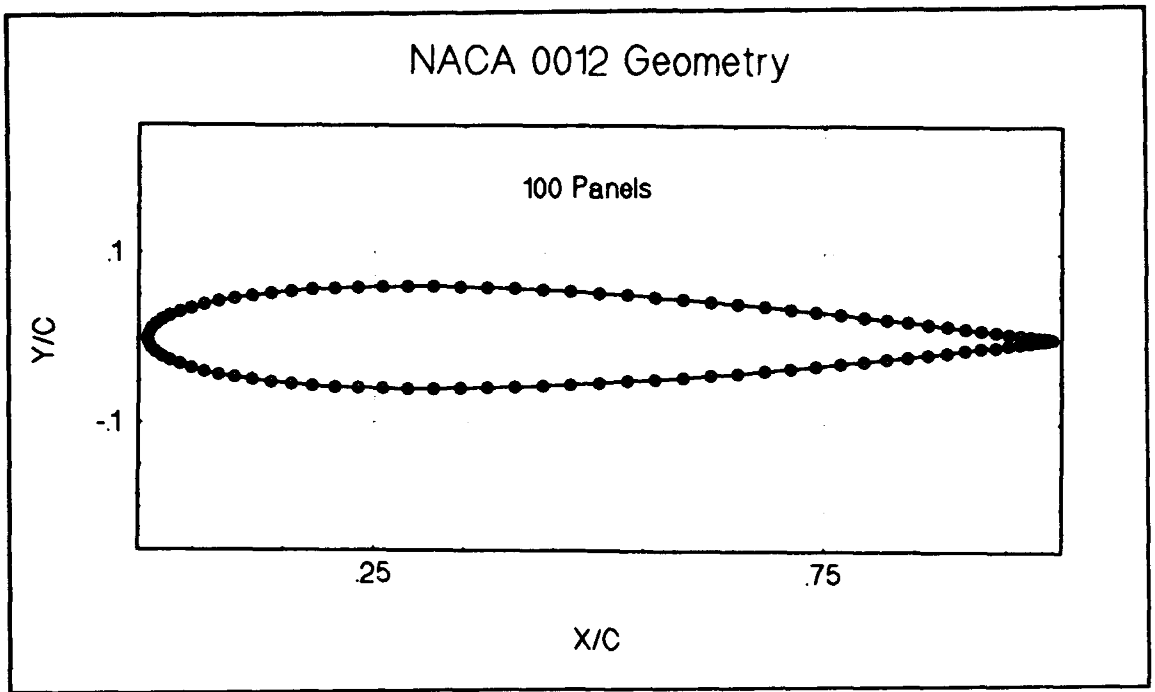


Figure 2.4

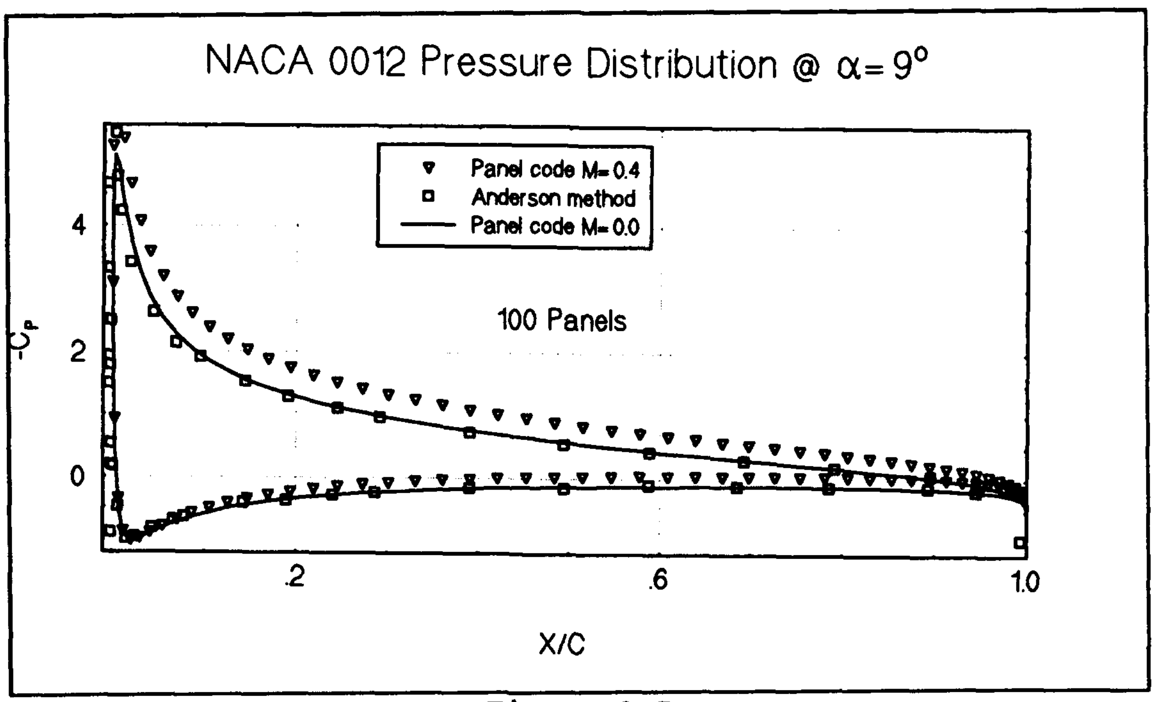


Figure 2.5

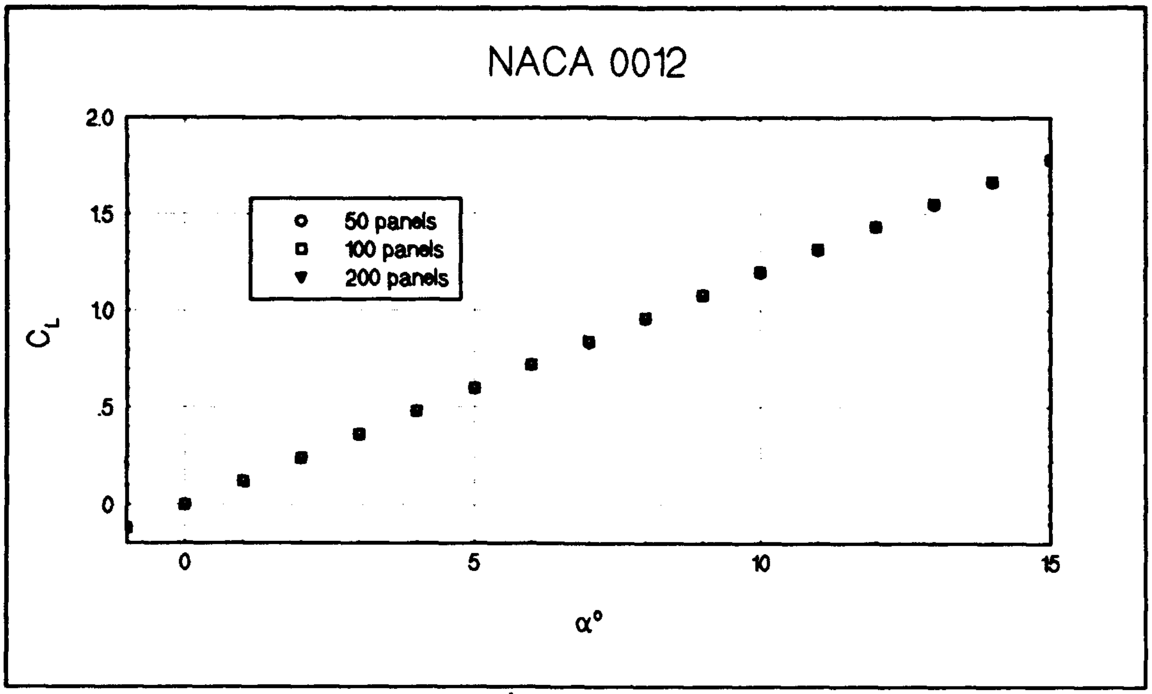


Figure 2.6

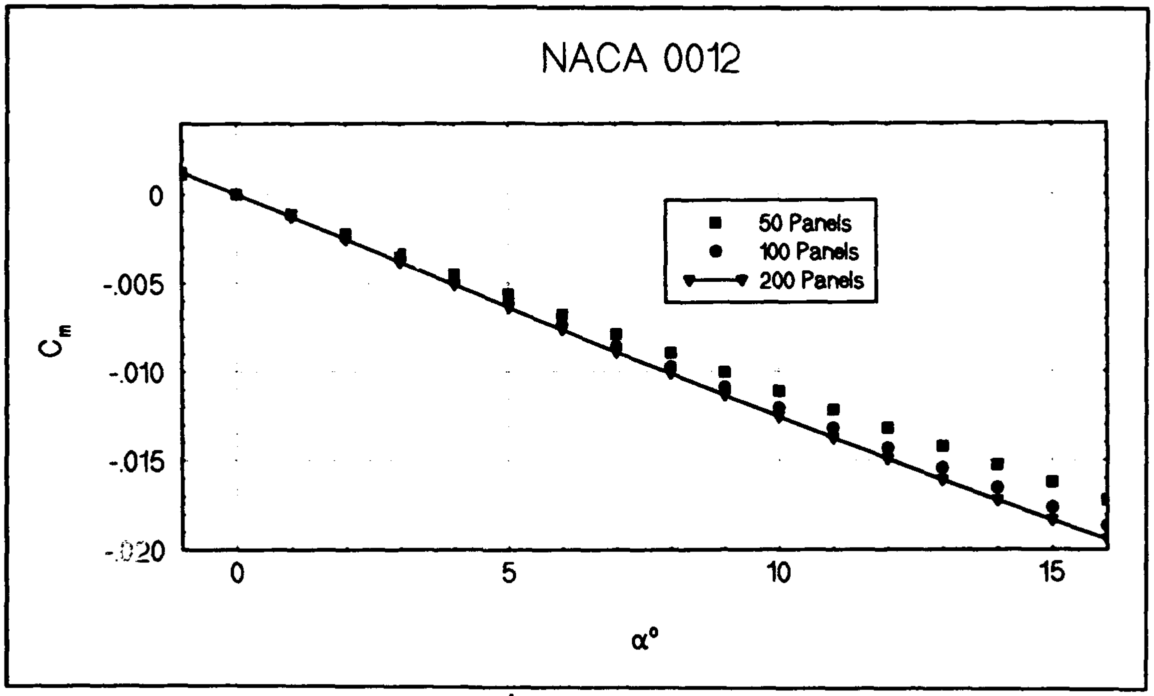


Figure 2.7

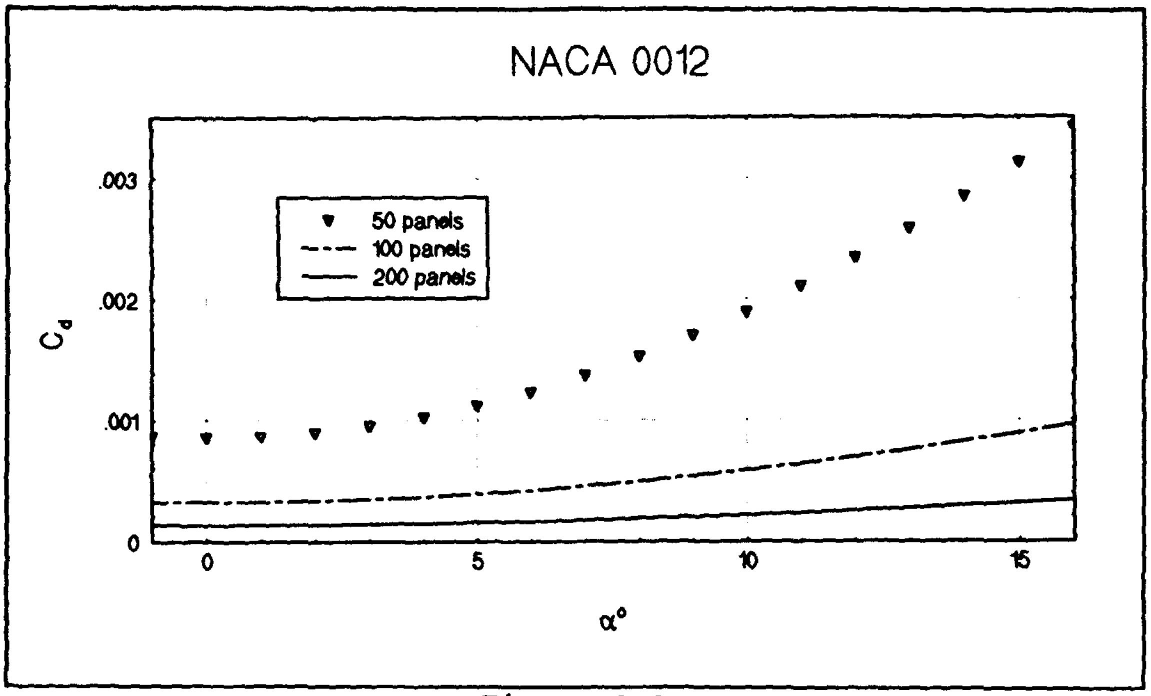


Figure 2.8

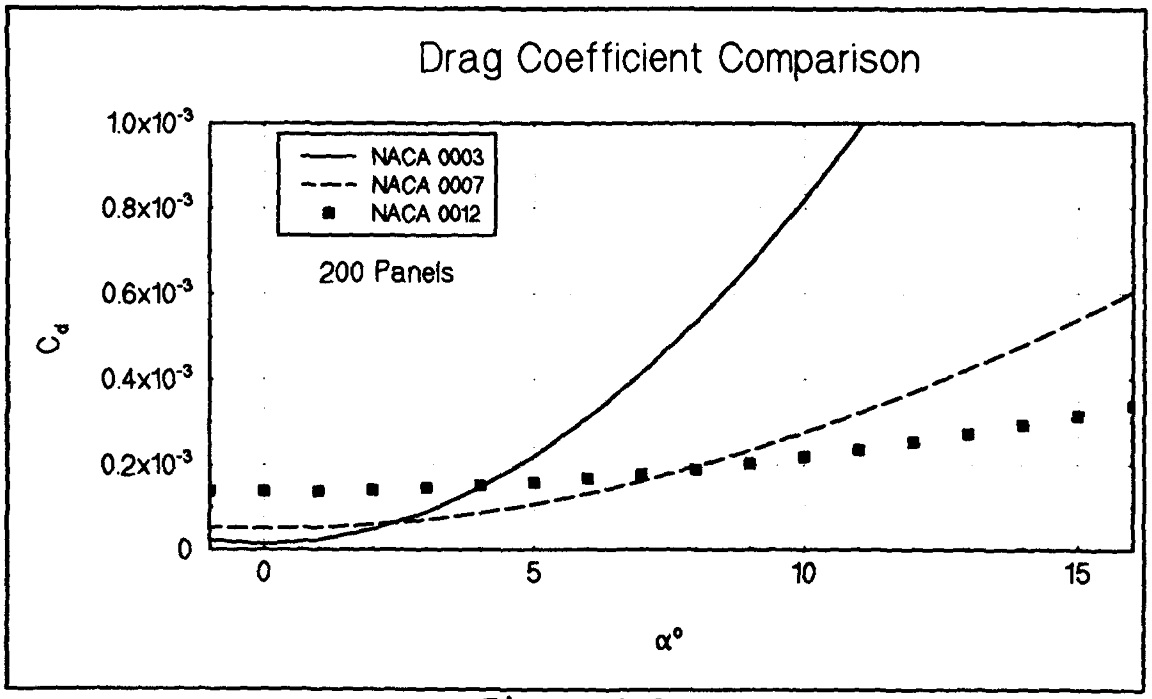


Figure 2.9

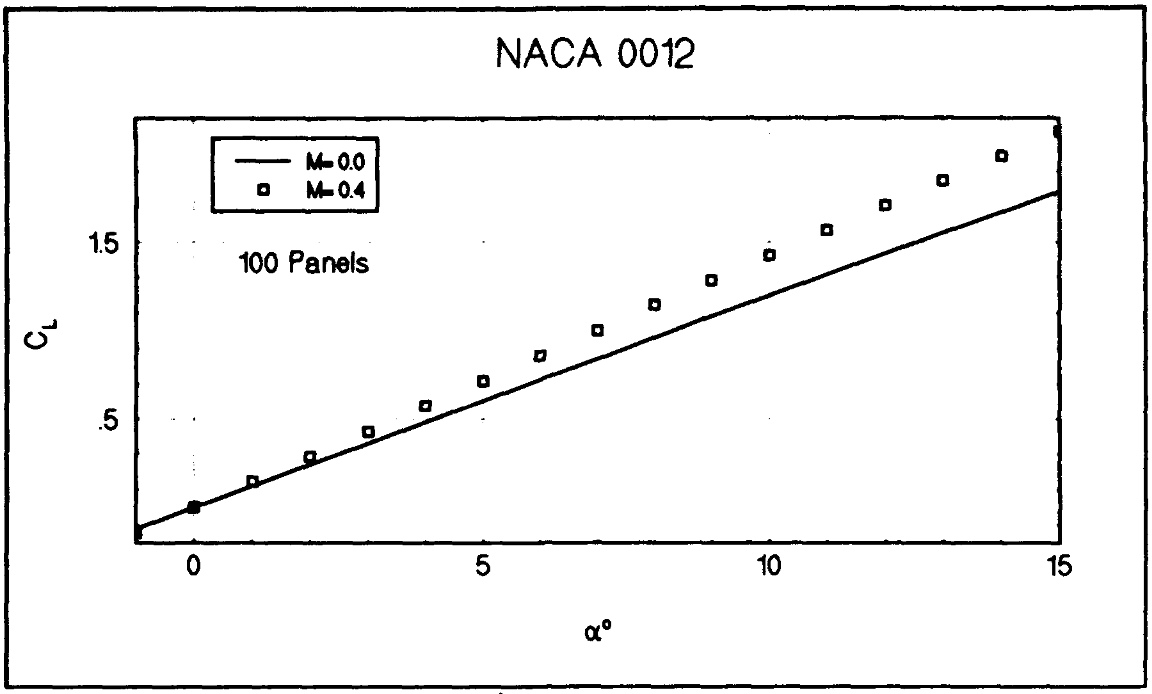


Figure 2.10

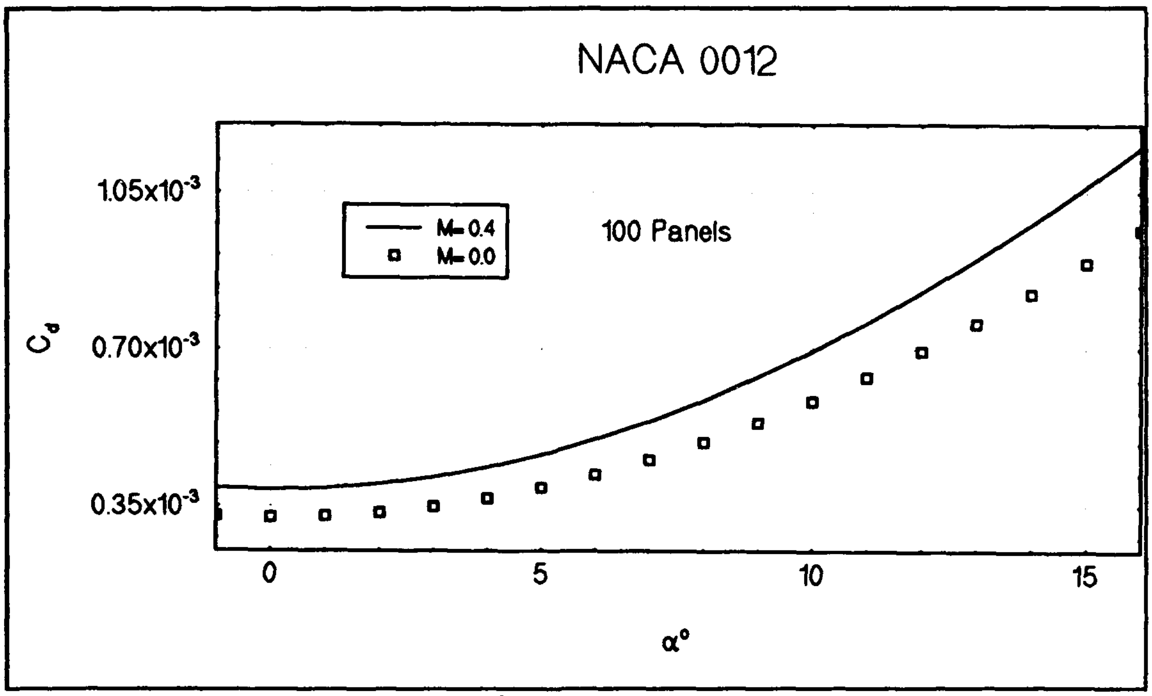


Figure 2.11

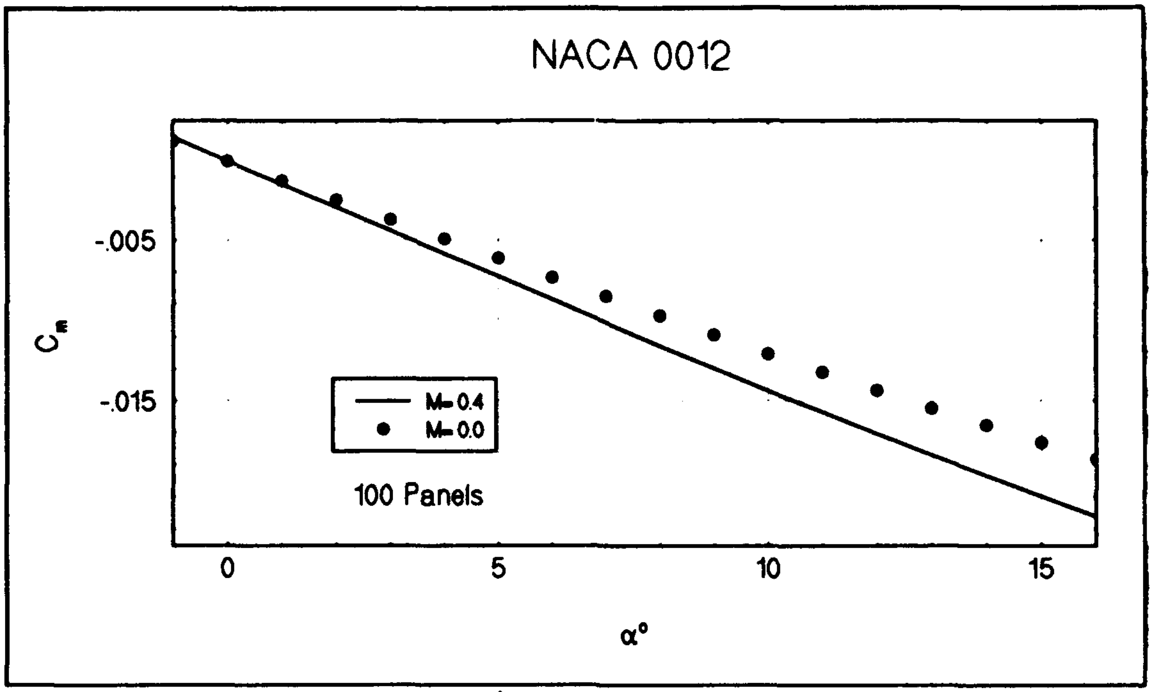


Figure 2.12

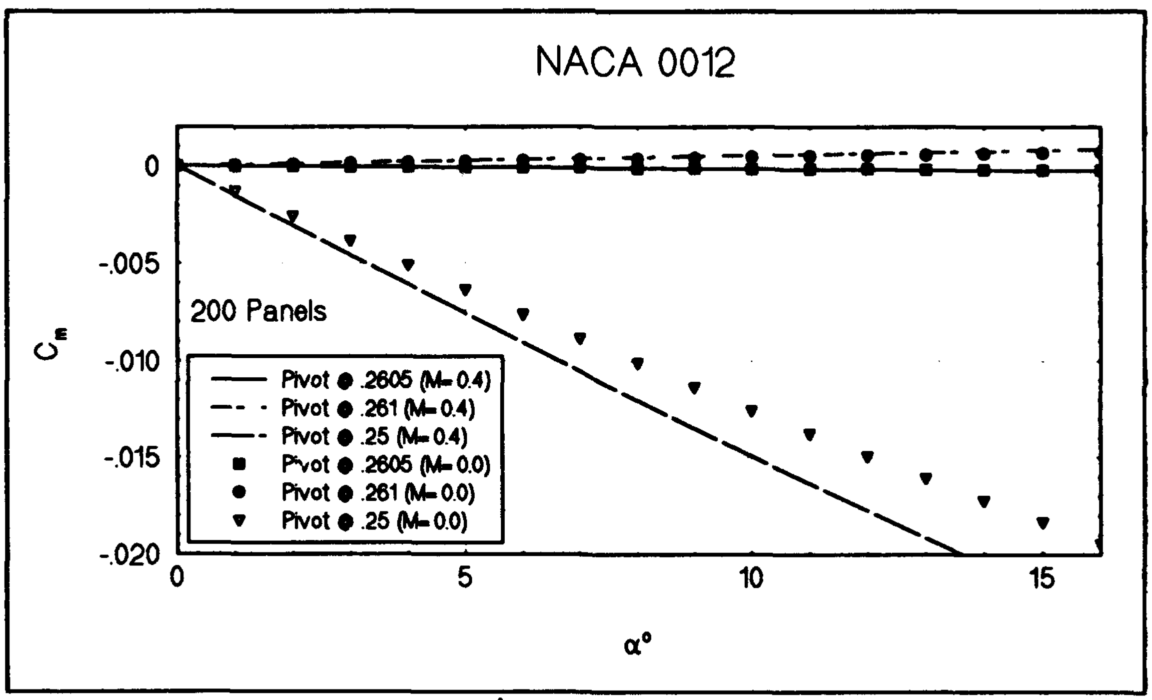


Figure 2.13

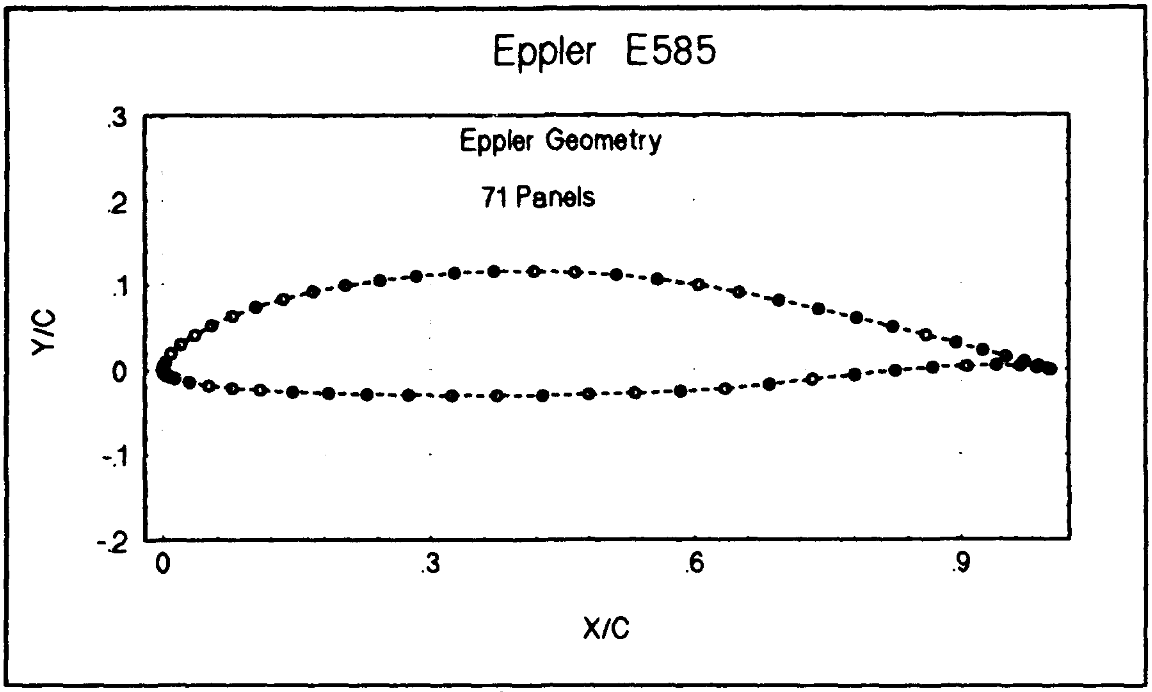


Figure 2.14

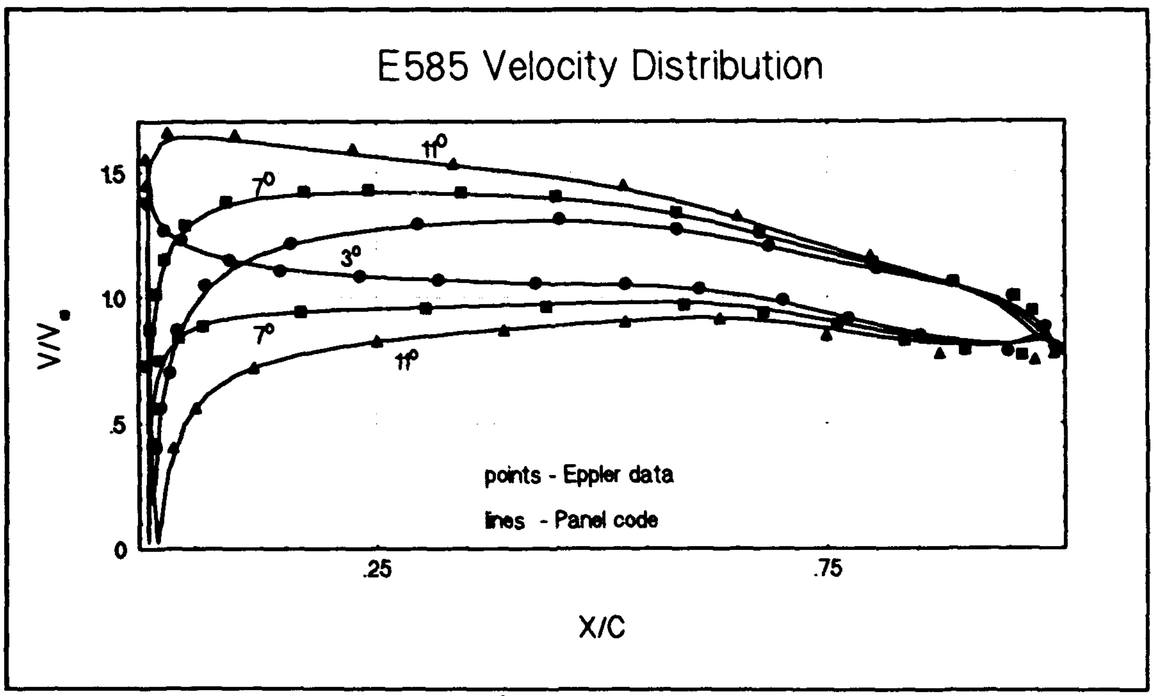


Figure 2.15

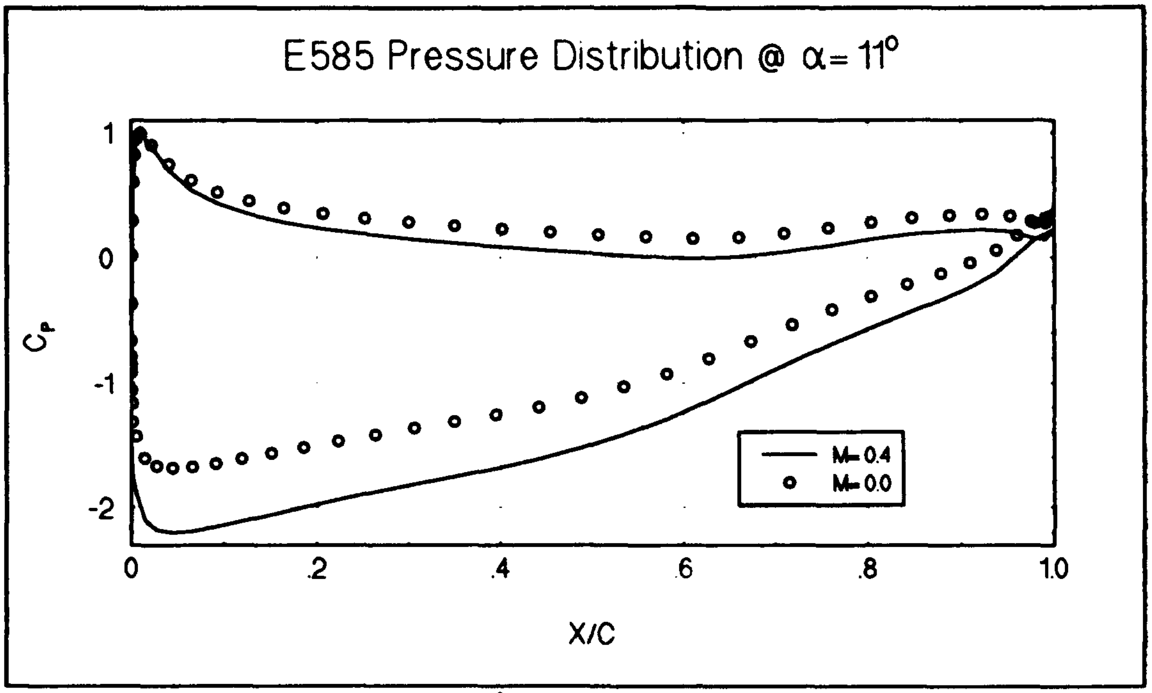


Figure 2.16

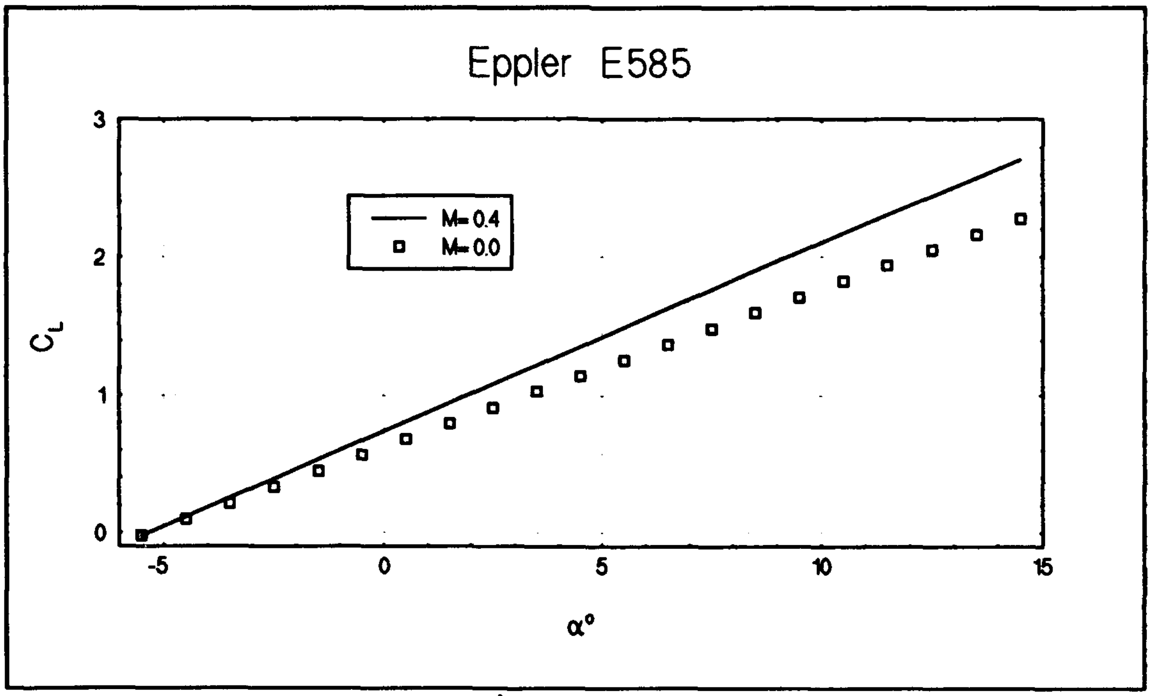


Figure 2.17

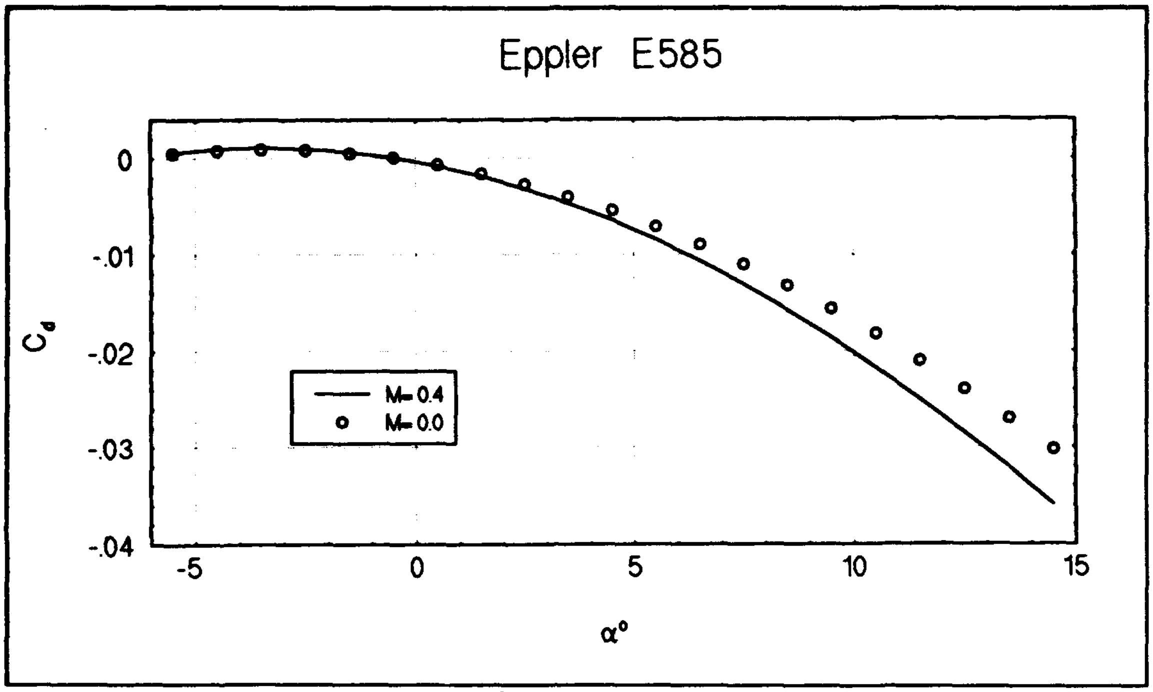


Figure 2.18

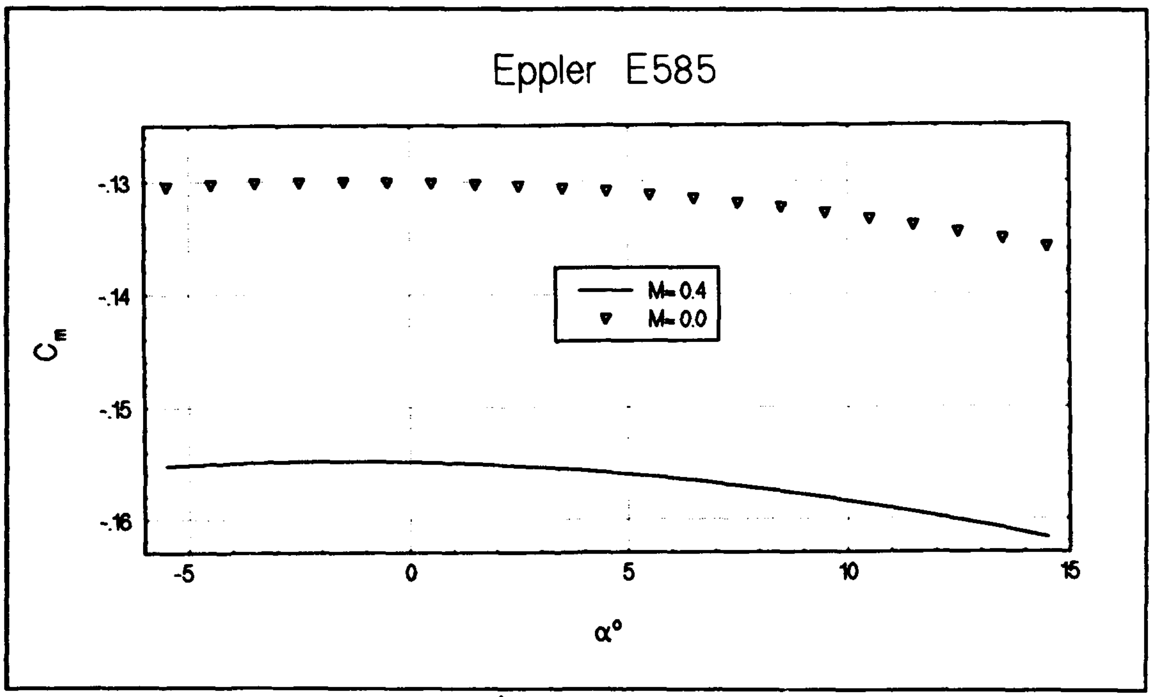


Figure 2.19

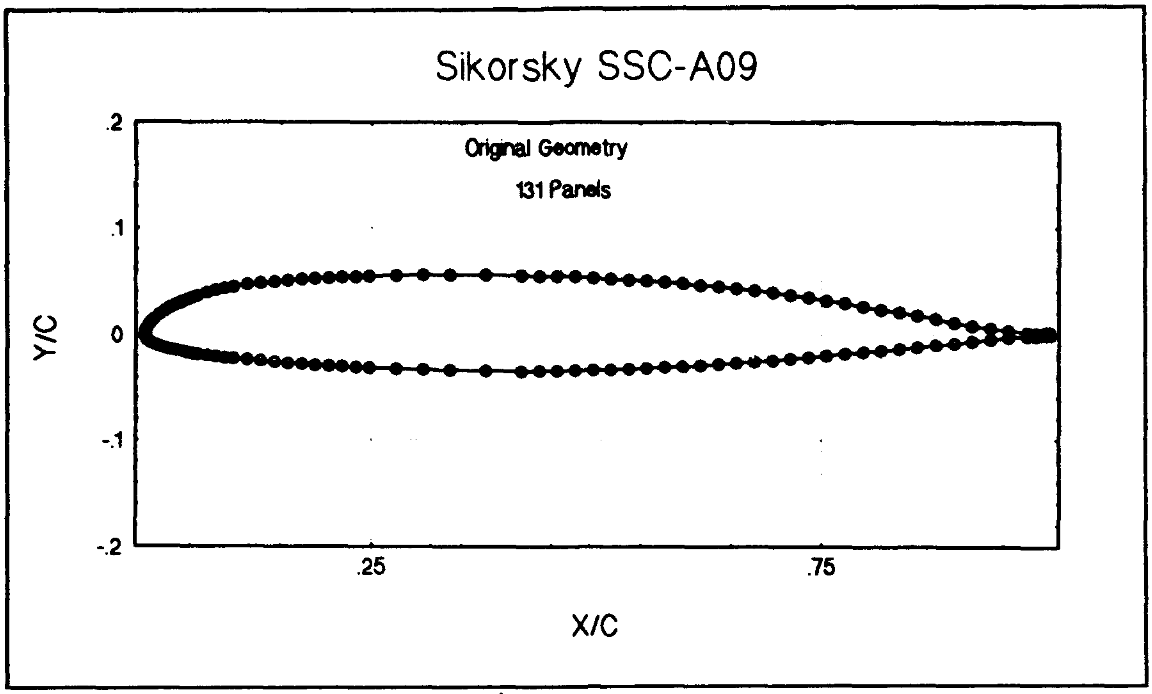


Figure 2.20

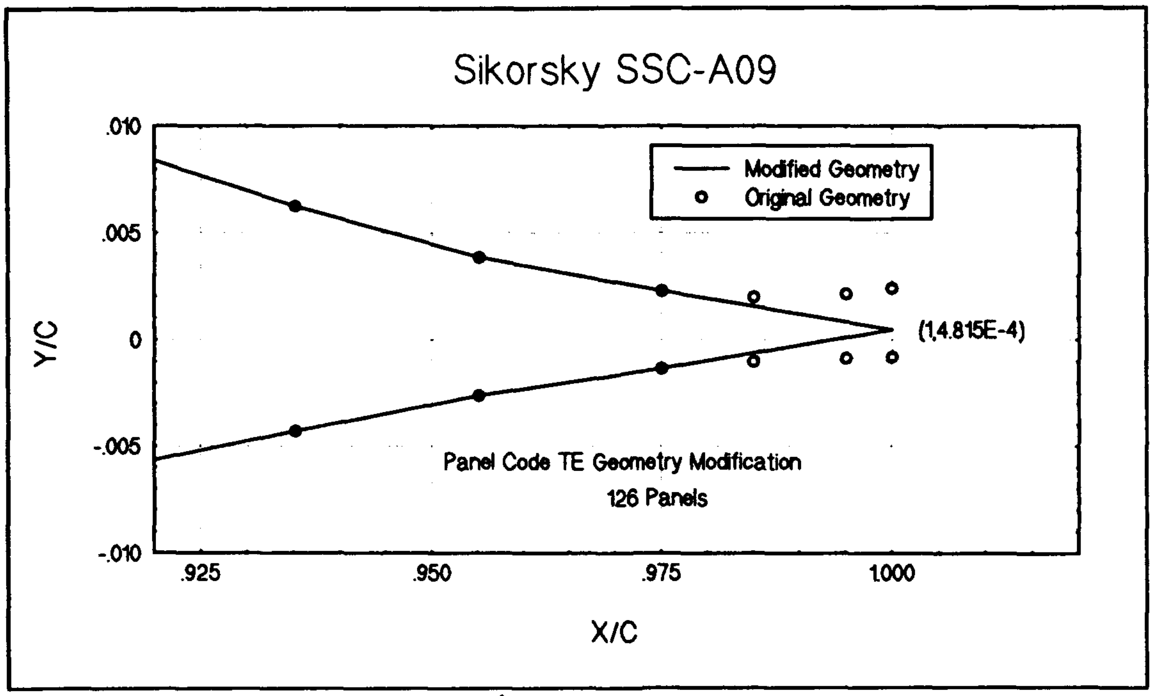


Figure 2.21

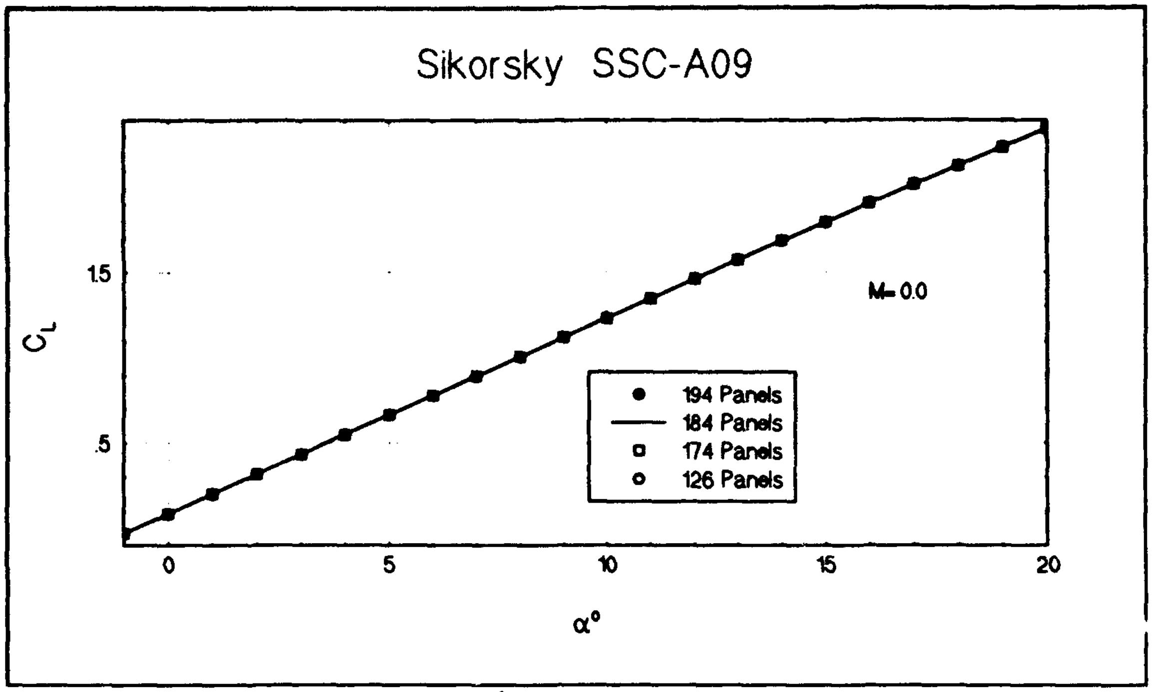


Figure 2.22

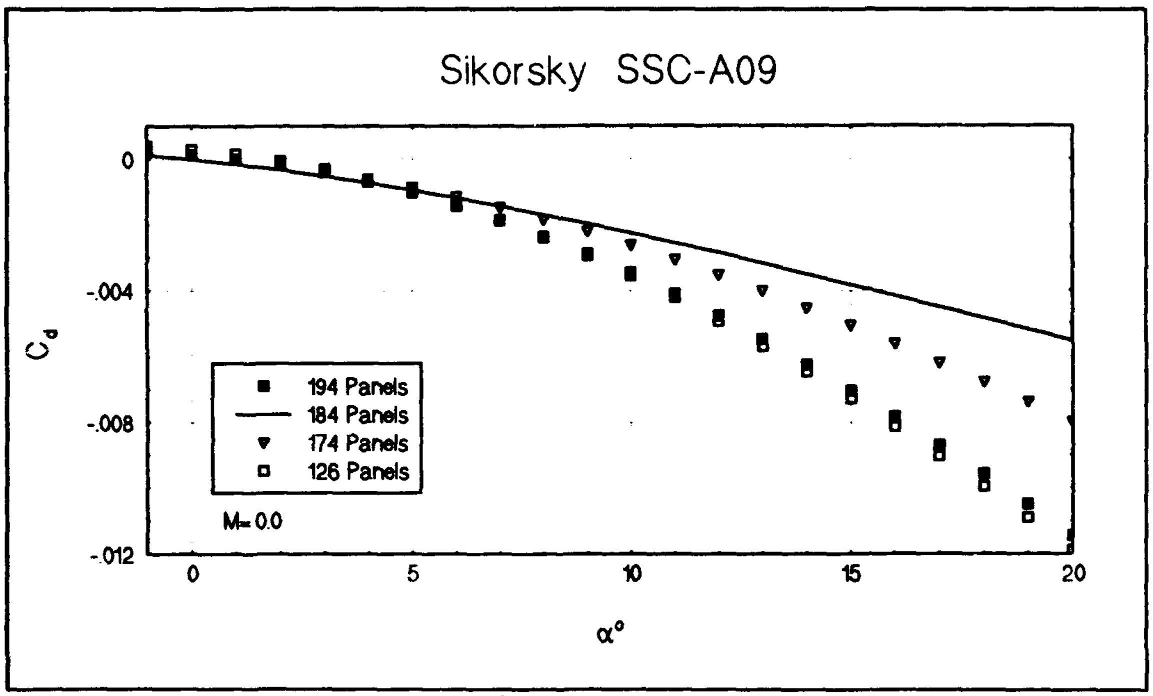


Figure 2.23

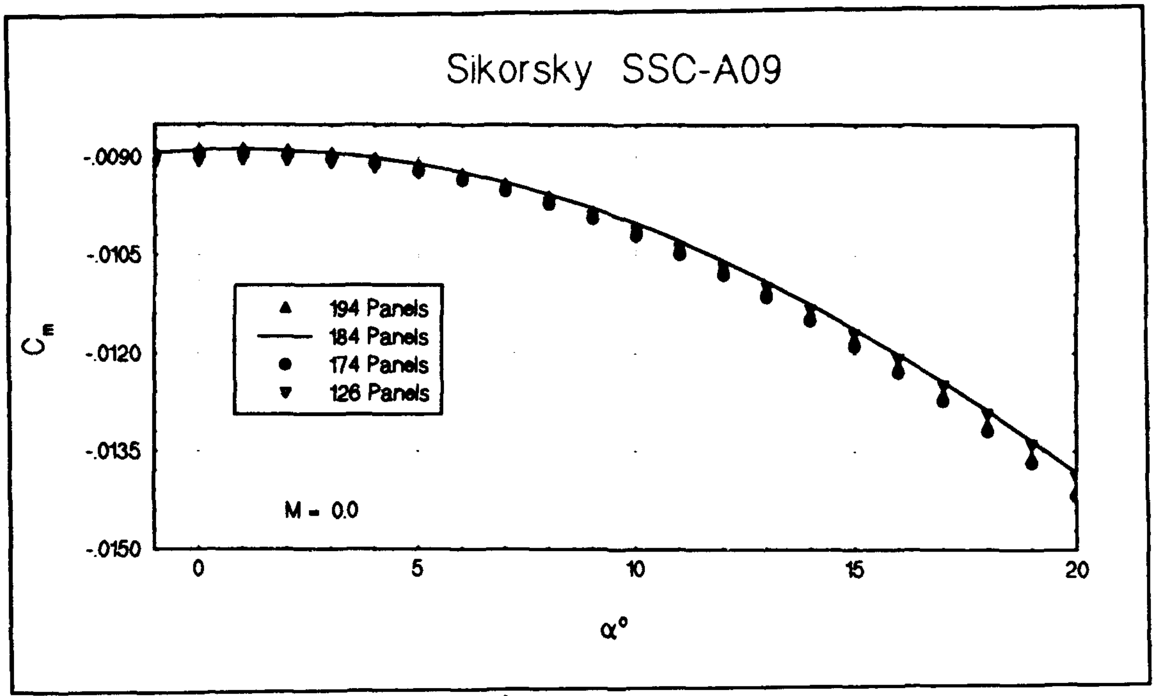


Figure 2.24

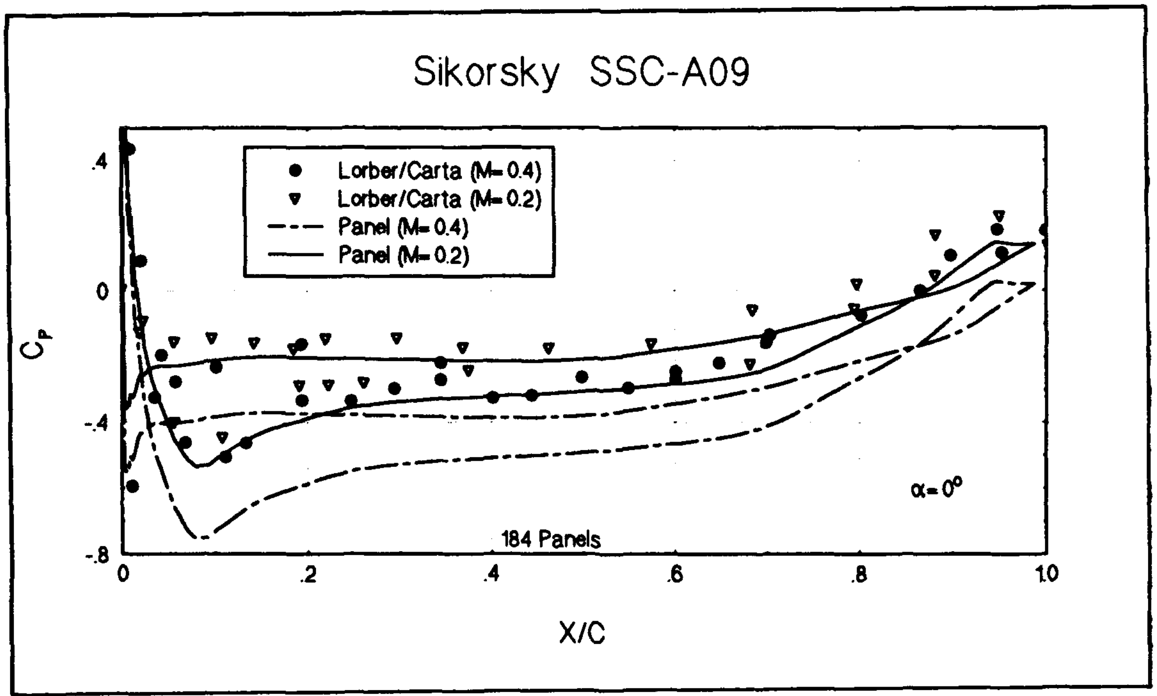


Figure 2.25

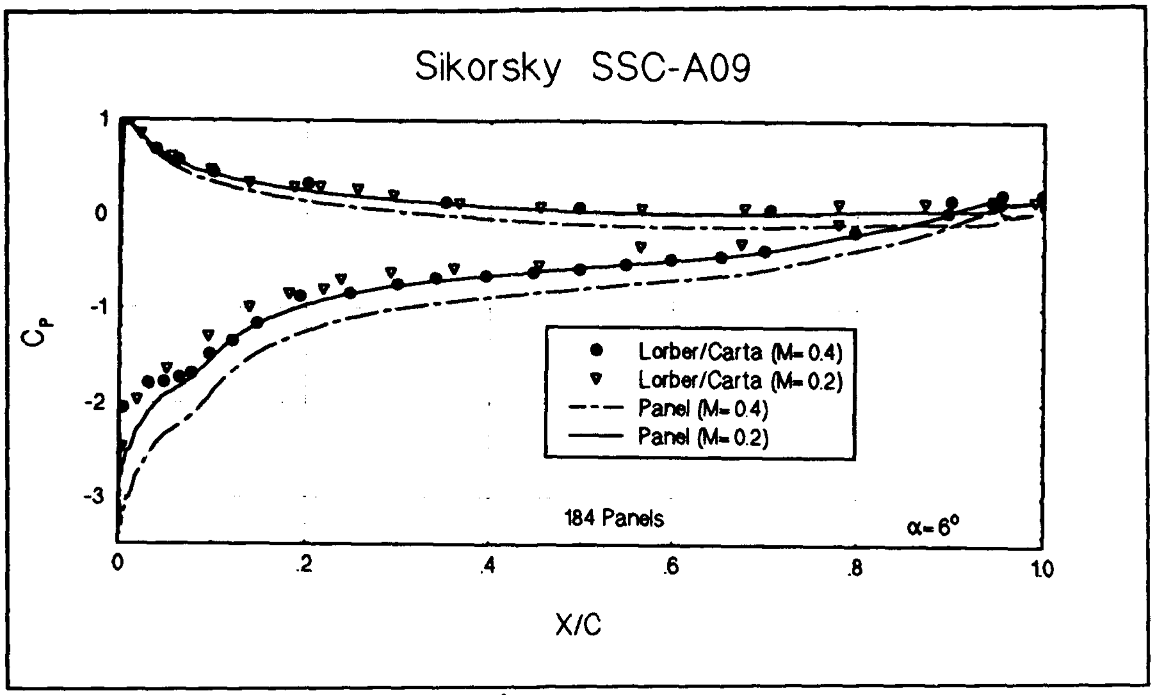


Figure 2.26

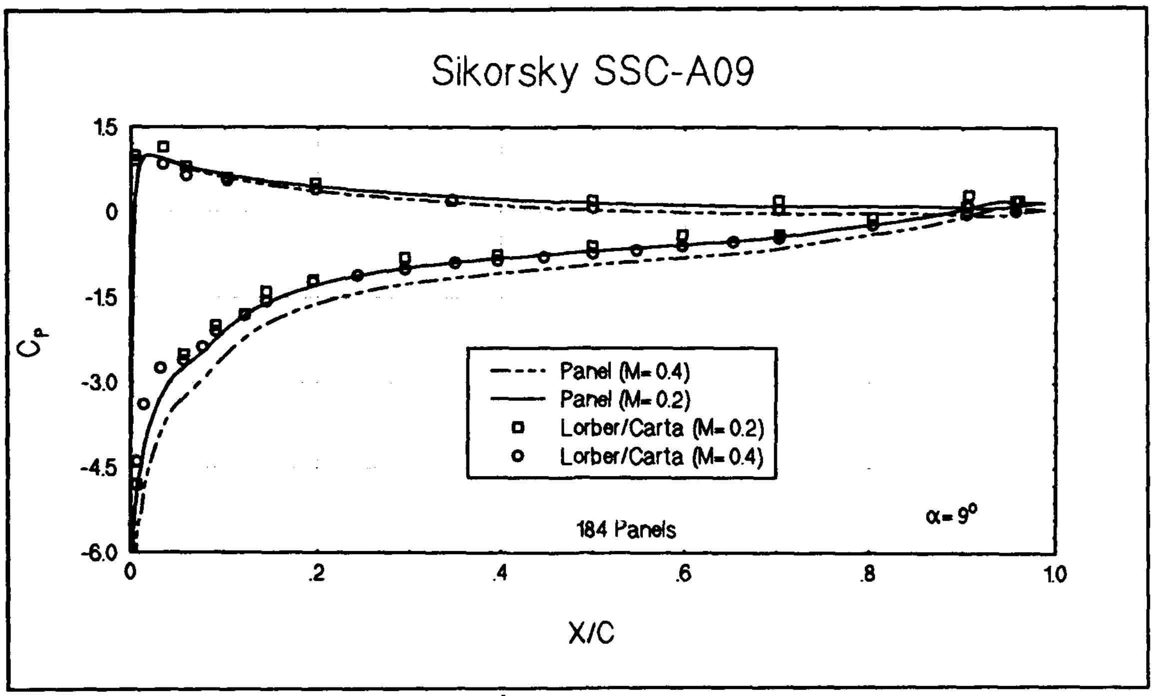


Figure 2.27

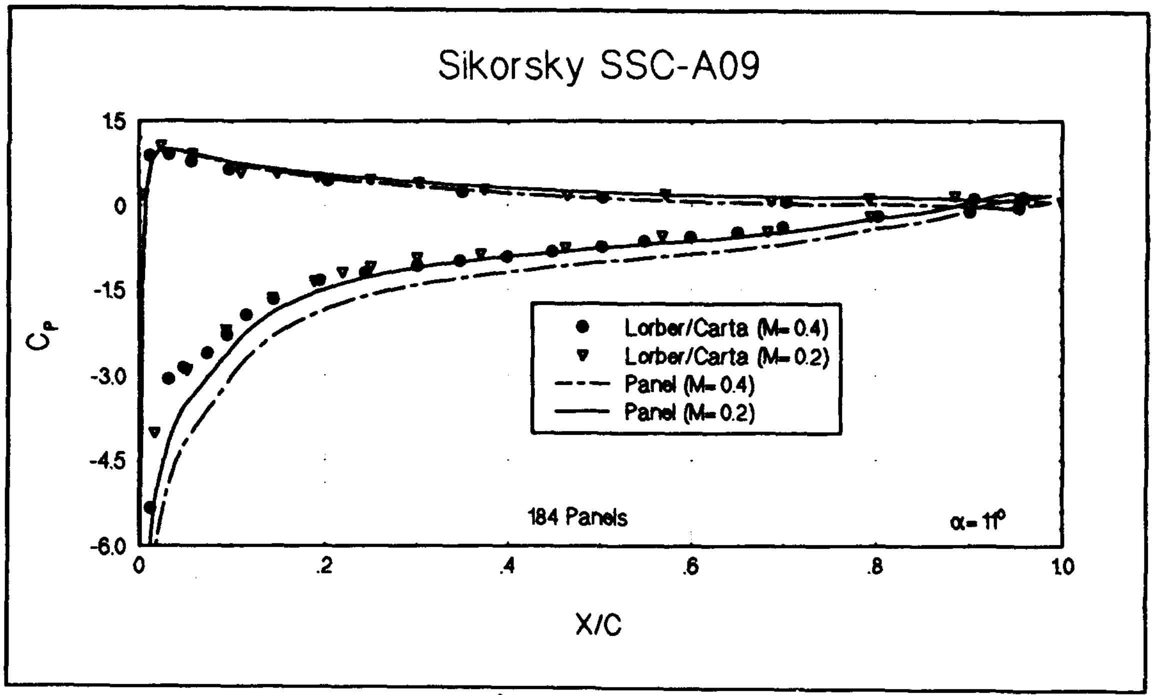


Figure 2.28

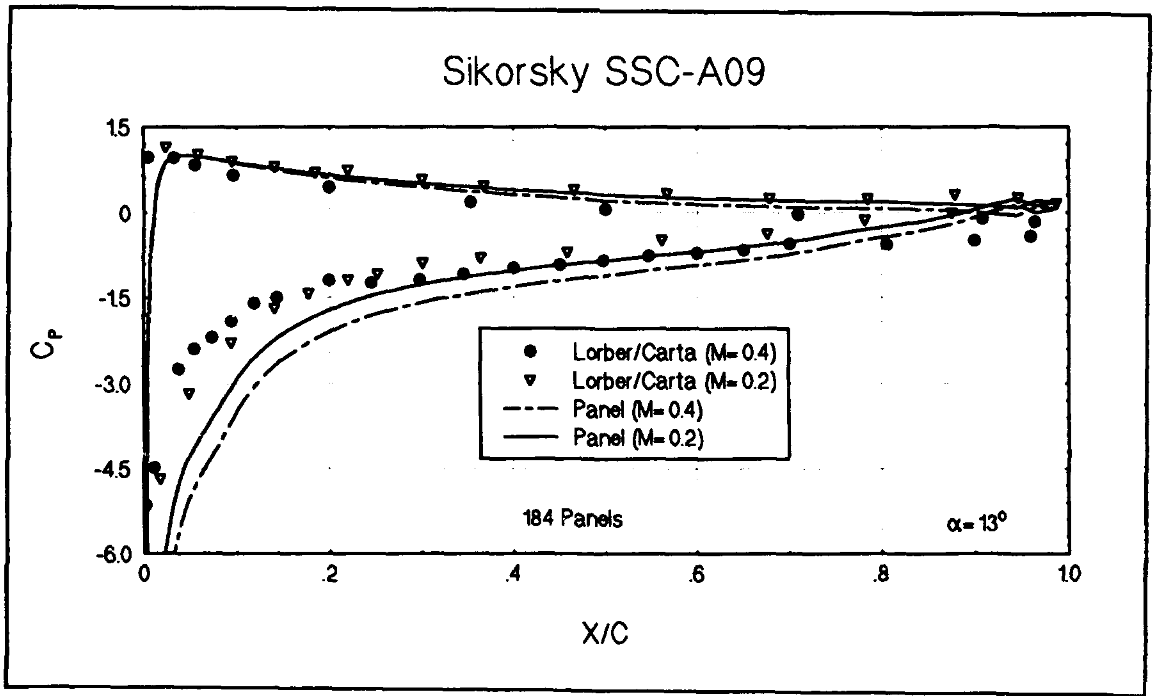


Figure 2.29

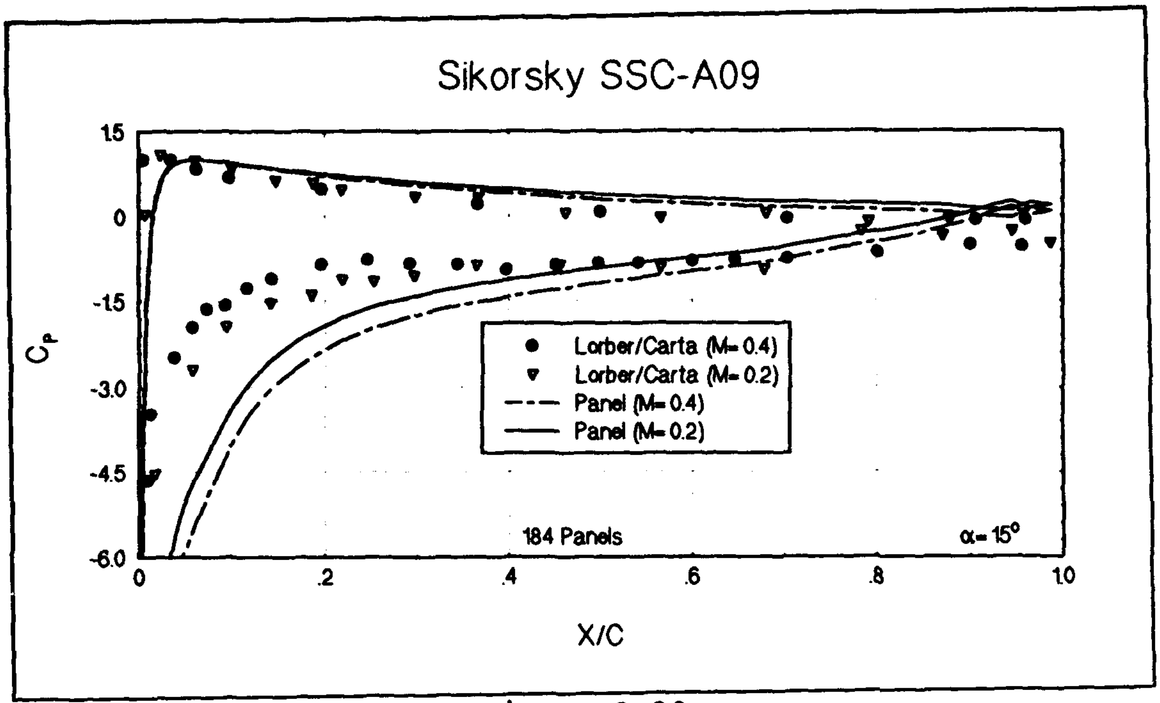


Figure 2.30

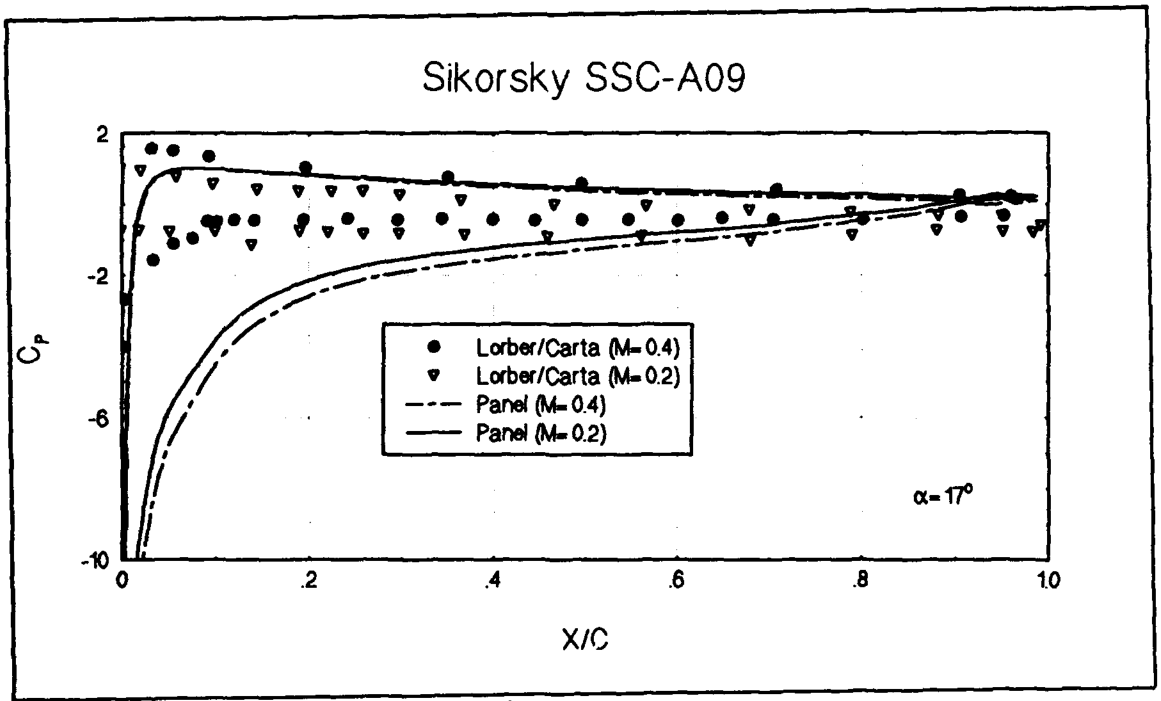


Figure 2.31

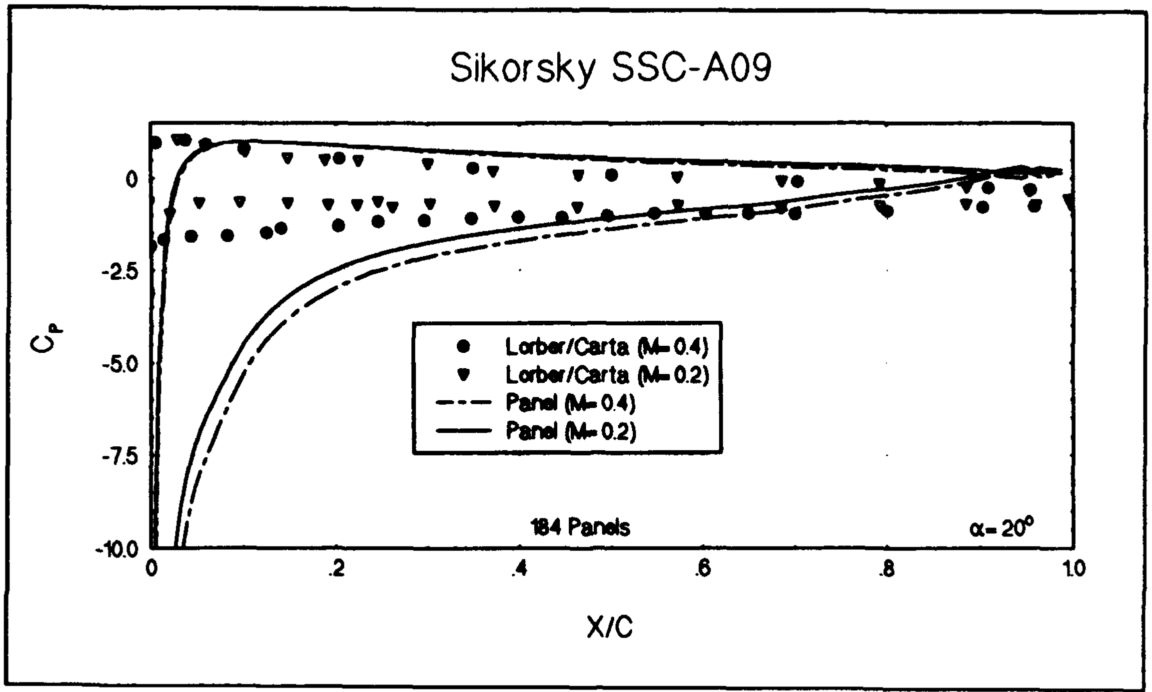


Figure 2.32

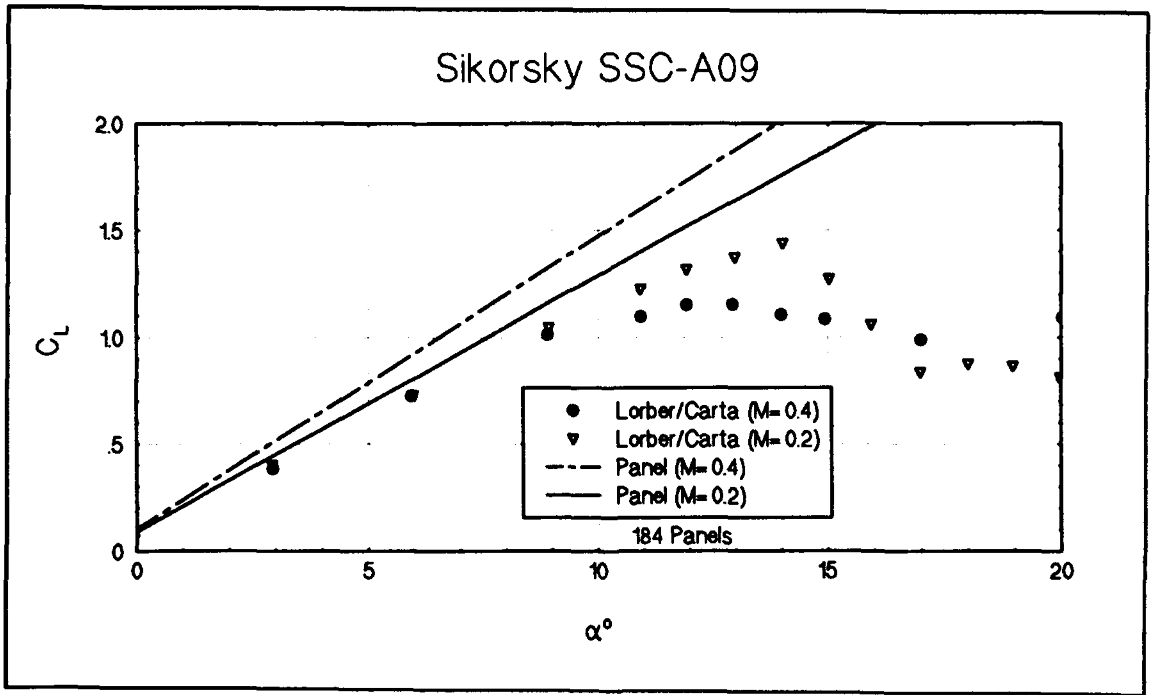


Figure 2.33

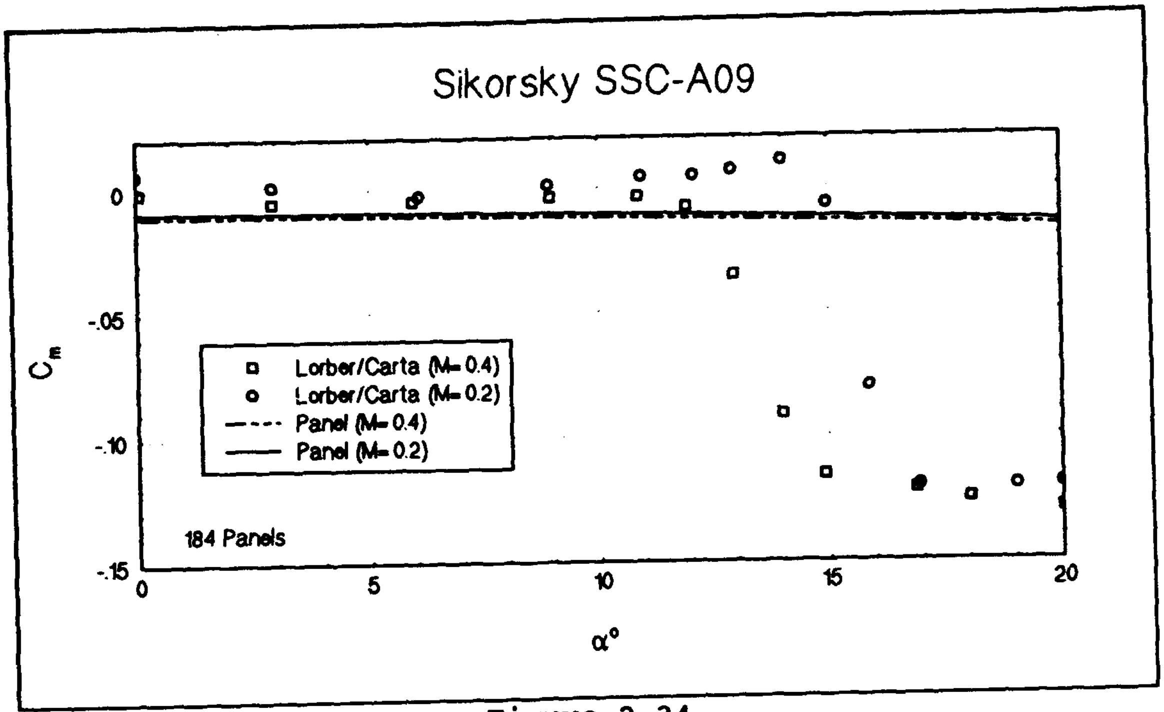


Figure 2.34

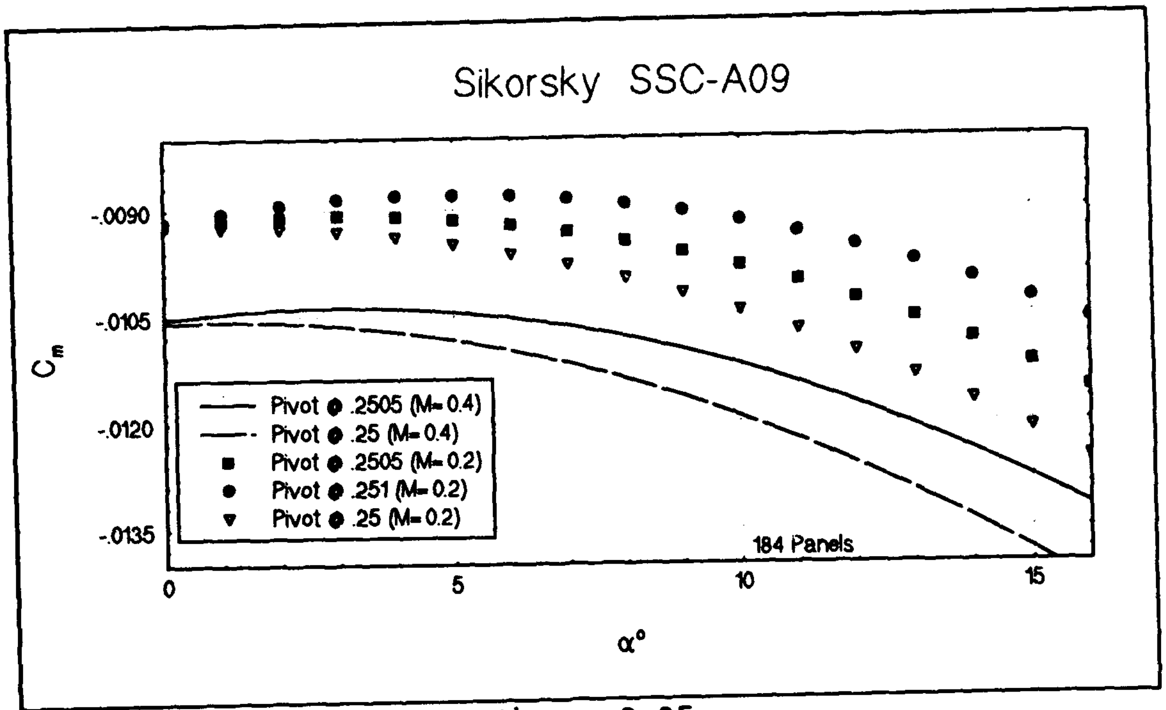


Figure 2.35

III. CEBECI 2-D LAMINAR & TURBULENT BOUNDARY LAYER CODE

A. SHEAR LAYER THEORY/BACKGROUND

Limiting the flow field to two-dimensional with no body forces, the generalized Navier-Stokes equations can be obtained when Newton's second law is applied to a finite control volume fixed in space or to an infinitesimally small moving fluid element. The resulting general unsteady, compressible, viscous flow Navier-Stokes equations become:

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} \quad (3.1)$$

$$\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho uv)}{\partial x} + \frac{\partial(\rho v^2)}{\partial y} = -\frac{\partial p}{\partial y} + \frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} \quad (3.2)$$

The unsteady continuity equation results when the conservation of mass principle is applied to a finite control volume fixed in space and is shown in Equation 3.3:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} = 0 \quad (3.3)$$

Since a complete flow field solution to these equations requires a vast amount of computer time and power, a dimensional/order-of-magnitude reduction of the Navier-Stokes equations results in the boundary layer equations. These equations allow a practical scheme to computationally solve the flow field.

When a steady, incompressible flow field is assumed, the energy equation will decouple from the momentum and continuity equations allowing ease of solution. Assuming that the fluid behaves as a Newtonian fluid, where viscous stress is proportional to the rate of fluid strain, assuming constant flow properties, and subtracting out the continuity equation the x-component of the momentum equation becomes:

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right] \qquad (3.4)$$

where ν is the kinematic viscosity ($\nu = \mu/\rho$).

There are three types of fluid momentum transfer:

- Transport by fluid mean motion.
- Transfer of random molecular motion (viscous stresses).
- Transfer by turbulent eddies (mean turbulent stresses).

Except at very low Reynolds numbers, viscous stresses are small compared to the rate of momentum transfer by the mean fluid element motion. Instantaneous flow quantities are replaced by a mean and fluctuating term to incorporate turbulent flow effects. This results in extra stress terms often called Reynolds stresses.

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right] - \frac{\partial \overline{u'^2}}{\partial x} - \frac{\partial \overline{u'v'}}{\partial y} \qquad (3.5)$$

The basic boundary layer theory assumptions are that the boundary layer is very thin when compared to the body length scale (airfoil chord), and the flow Reynolds number is large. An order-of-magnitude analysis of the x and y turbulent flow momentum equations result in the steady, two-dimensional, incompressible **Boundary Layer Equations** for laminar and turbulent flows (Cebeci and Bradshaw [Ref. 5]):

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \qquad (3.6)$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p_{ext}}{\partial x} + \nu\frac{\partial^2 u}{\partial y^2} - \frac{\partial\,\overline{u'v'}}{\partial y} \qquad (3.7)$$

$$\frac{\partial p}{\partial y} = 0 \qquad (3.8)$$

The y-component of the momentum equation implies that pressure is constant through the boundary layer in the direction normal to the surface. This means that the pressure distribution at the boundary layer outer edge is impressed directly onto the surface without change. This assumption is generally true as long as one stays away from large curvatures (Anderson [Ref. 2]). This allows division of the flow field into an inner, the viscous boundary layer region, and an outer region where viscous stresses are negligible and thus can be treated and solved using incompressible, inviscid numerical methods. Two boundary conditions are applied. The no slip,

airfoil surface boundary condition is represented by **u=v=0**. The outer boundary layer edge condition is **$y=\delta$** at **$U=U_e(x)$**.

1. Turbulence Model

Turbulence modeling is used to relate the Reynolds shear stress term of Equation 3.7 to the local mean-velocity gradient allowing numerical flow field calculation. This modeling is based on local equilibrium - the assumption that the transport terms are small. Prandtl proposed a mixing length model, Equation 3.9, similar to the kinetic theory of gases where turbulent eddies are assumed to be discrete and to collide and exchange momentum at distinct/discrete intervals. Here, l is a characteristic length related to the fluid turbulence intensity (Cebeci and Bradshaw [Ref. 5]).

$$-\rho\ \overline{u'v'} \;=\; \rho l^2 \left|\frac{\partial u}{\partial y}\right| \frac{\partial u}{\partial y} \tag{3.9}$$

Boussinesq proposed a mean flow, eddy-viscosity model, Equation 3.10, where ϵ_m is termed the turbulent eddy-viscosity and is assumed to vary less rapidly than the shear stress term. It is important to note that eddy-viscosity is not a flow property and depends greatly on the mean-velocity gradient and mixing length (Cebeci and Bradshaw [Ref. 5]).

$$-\rho\ \overline{u'v'} \;=\; \rho\ \epsilon_m \frac{\partial u}{\partial y} \tag{3.10}$$

The Cebeci-Smith eddy-viscosity model, Equations 3.11 and 3.12, is used for separated flow computation and treats

the boundary layer as a composite layer having an inner, ϵ_{mi}, and an outer, ϵ_{mo}, region with separate empirical formulations. The inner eddy-viscosity defines the region from the airfoil surface outward until $\epsilon_{mi}=\epsilon_{mo}$, where the outer eddy-viscosity takes over to the edge of the boundary layer (Cebeci and Bradshaw [Ref. 5]).

$$\left(\frac{\epsilon_m}{\nu}\right)_i = .16\ R_{e_x}^{\frac{1}{2}} \left[1 - e^{-\left(\frac{y}{A}\right)}\right]^2 \eta^2\ v\ \gamma_{tr} \qquad (3.11)$$

$$\left(\frac{\epsilon_m}{\nu}\right)_o = .0168\ R_{e_x}^{\frac{1}{2}}\ [\eta_\infty - f_\infty]\ \gamma_{tr} \qquad (3.12)$$

$$\begin{aligned}
R_{e_x} &= \frac{U_e}{U_\infty}\ \xi\ R_L \\
\frac{y}{A} &= \frac{1}{26}\ R_{e_x}^{\frac{1}{4}}\ v_w^{\frac{1}{2}}\ \eta \\
\gamma_{tr} &= 1 - \exp\left[-G\ (x-x_{tr}) \int_{x_{tr}}^{x} \frac{dx}{U_e}\right] \qquad (3.13) \\
G &= \frac{1}{1200}\left[\frac{U_e}{U_\infty}\right]^3 R_L^2\ R_{e_{x_{tr}}}^{-1.34} \\
f(x,\eta) &= \frac{\Psi(x,y)}{\sqrt{u_e \nu x}}
\end{aligned}$$

2. Transition Model

Laminar to turbulent flow transition presents a stability problem where vortical interaction is very non-linear. The Chen-Tyson transition model utilizes a region of intermittency that is controlled by the intermittency factor,

γ_{tr}, which allows turbulence to gradually build in the streamwise direction creating a transition zone instead of a laminar to turbulent transition point.

Michel's empirical correlation curve for transition, Equation 3.14, is used as an initial estimate for transition location. It is based on incompressible and constant property flow. See Appendix D for further discussion.

$$R_{\theta_{tr}} = 1.174 \left[1 + \frac{22,400}{R_{e_{x_{tr}}}}\right] R_{e_{x_{tr}}}^{.46}$$

$$R_{\theta_{tr}} = \frac{U_e \, \theta}{\nu} \tag{3.14}$$

$$R_{e_{x_{tr}}} = \frac{U_e \, x}{\nu}$$

B. BL2D.F OVERVIEW

A two-dimensional, steady, incompressible, viscous flow program was developed by Cebeci and Bradshaw [Ref. 5] to provide solutions to the (Thin Shear Layer) boundary layer equations using the Cebeci-Smith eddy-viscosity turbulence model and the Chen-Tyson transition model. Required inputs for operation are:

- An external velocity distribution.
- Airfoil surface coordinates.
- Flow Reynolds number.
- A natural transition point estimate (upper and lower).

- The forward stagnation point location.

The program unwraps the surface coordinate onto the x-axis. A Falkner-Skan variable transformation is made to analyze laminar boundary layers and to reduce the turbulent boundary layer growth. The transformed coordinates are nearly independent in the streamwise direction. The Keller-Cebeci box, Newton's, and block tridiagonal methods are used to solve the second order partial differential equations. The program generates output files for graphical visualization and interpretation:

- Skin friction coefficient
- Displacement thickness
- Boundary layer velocity profiles

The laminar, transitional, and turbulent boundary layers are calculated starting from the forward stagnation point. A complete users guide for BL2D.F is located in Appendix A.

1. Program Hints

Convergence is **critically dependent** on the upper surface transition point input and to a lesser degree on the forward stagnation point input. If the laminar flow calculations indicate flow separation (a separation bubble) before the transition point can be calculated, the wall shear becomes negative causing solution divergence and meaningless results.

Transition from laminar to turbulent flow is indicated where C_f reaches a minimum and then dramatically increases. Separation is indicated when C_f reaches zero or a negative value. Nowak [Ref. 13] discovered that the program can handle mild amounts of separation with a symmetric airfoil at angle-of-attack. She also found that increasing Reynolds number decreases the probability of separation.

C. THE NACA 0012 AIRFOIL

The boundary layer code, bl2d.f, was validated using previously documented results. A test case was completed at a Reynolds number of one million and a transition location of .38 X/C and presented in Figure 3.1. Skin Friction Coefficient, C_F, and Displacement Thickness, δ^*, results agree well with those presented in Cebeci and Bradshaw [Ref. 5].

Nowak [Ref. 13] presented results for this airfoil at a Reynolds number of 540,000. BL2D.F results are shown in Figures 3.2 through 3.7. Also presented are the variations in C_F and δ^* outputs as a function of computer type: Indigo and Stardent. Table 3.1 presents the various input parameters. Variation in C_F and δ^* outputs appear dependent on how the specific computer handles C_F as it approaches zero or becomes negative. Michel's empirical estimate also varied between computers, depending on input (See Appendix D). As expected, the transition point moves forward on the upper surface as angle-of-attack (α^0) increases. The Stardent computed a

slight separation zone at 4° α located at 0.25 X/C and at 6° α located at 0.05 X/C when the Indigo did not. Both computers calculated leading edge suction separation bubbles at 10° α but at slightly different positions. Neither computer could arrive at a converged solution at greater than 10° α.

TABLE 3.1
NACA 0012 (100 PANELS)
INCOMPRESSIBLE FLOW AT R_e=540,000
BOUNDARY LAYER TRANSITION INPUTS

		Indigo			Stardent		
α°	Calculated Stagnation Point	Input Stagnation Point	Michel's Estimate	Transition Point	Input Stagnation Point	Michel's Estimate	Transition Point
0	51	51	0.597	0.578	51	0.597	0.545
2	50	51	0.374	0.390	50	0.379	0.380
4	49	48	0.277	0.219	50	0.212	0.301
6	48	48	0.156	0.054	47	0.092	0.070
8	47	47	0.285	0.027	47	0.045	0.044
10	46	46	0.055	0.075	46	0.055	0.041

D. THE SIKORSKY SSC-A09 AIRFOIL

The boundary layer code was run using the SSC-A09 airfoil and calculated C_F and δ^* results as a function of Mach number are illustrated in Figures 3.8 through 3.13. Table 3.2 presents the various input parameters. Again, the general upper surface transition point trend is to move forward as angle-of-attack increases. Compressibility causes C_F and δ^* to be thinner over the angle-of-attack range; the difference

increasing with increasing X/C. Small separation bubbles can be observed at 4° α located at 0.1 X/C, 8° α located at 0.03 X/C, and 9° α located at 0.02 X/C.

TABLE 3.2
SIKORSKY SSC-A09
184 PANELS
BOUNDARY LAYER TRANSITION INPUTS

		R_e=2E6 (M=0.2)			R_e=4E6 (M=0.4)		
α°	Calculated Stagnation Point	Input Stagnation Point	Michel's Estimate	Transition Point	Input Stagnation Point	Michel's Estimate	Transition Point
0	94	94	0.438	0.65	94	0.027	0.068
2	92	92	0.184	0.63	92	0.146	0.064
4	90	89	0.121	0.49	89	0.115	0.060
6	86	86	0.105	0.075	86	0.079	0.075
8	82	81	0.0163	0.023	82	0.010	0.070
9	81	82	0.0160	0.022	81	0.011	0.022

Figures 3.14 through 3.25 present the upper surface boundary layer velocity profiles as a function of Mach number and angle-of-attack. Increasing Reynolds number appears to thicken the boundary layer profile placing more fluid energy closer to the airfoil surface and resulting in a decrease in the separation bubble size. An example of this can be seen in Figures 3.18 and 3.19 at 4° α. A large separation bubble appears in the 2,000,000 Reynolds number flow but is greatly reduced in the 4,000,000 Reynolds number flow. The effect can also be observed at 8° α in Figures 3.22 and 3.23.

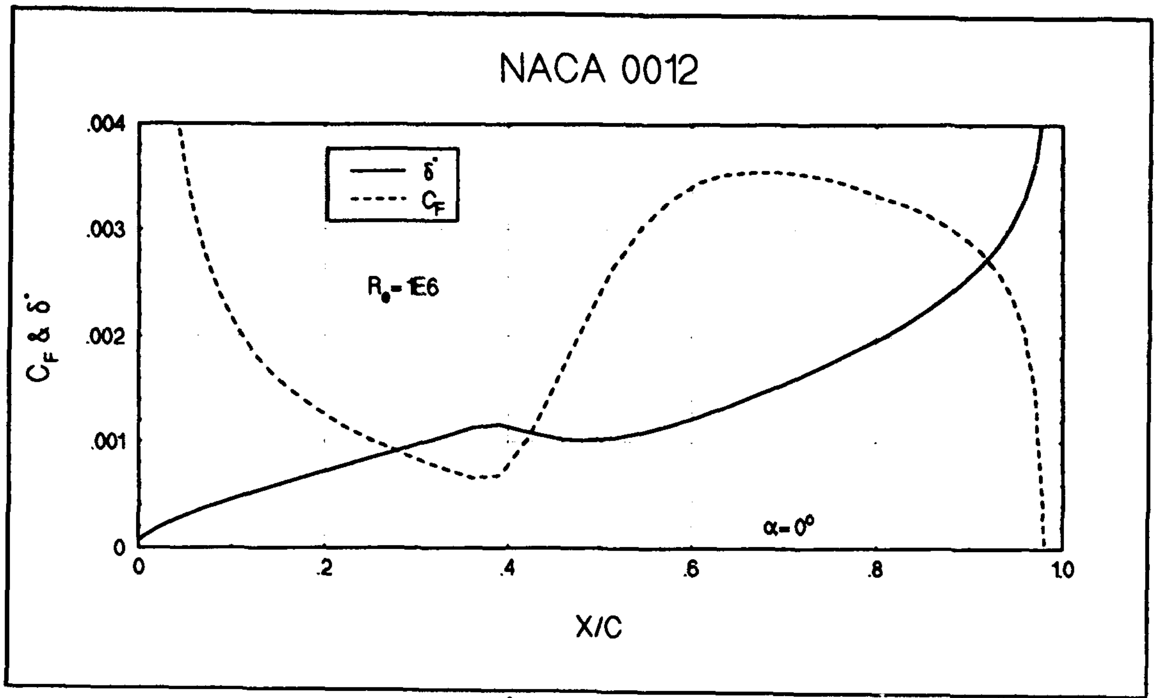


Figure 3.1

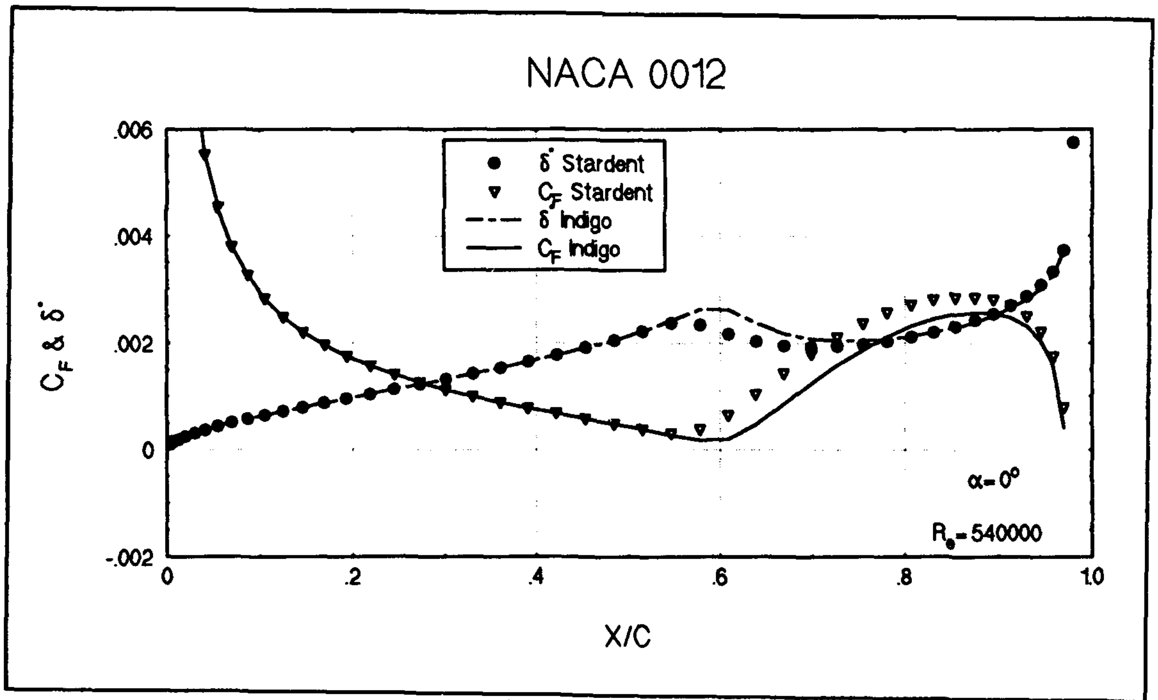


Figure 3.2

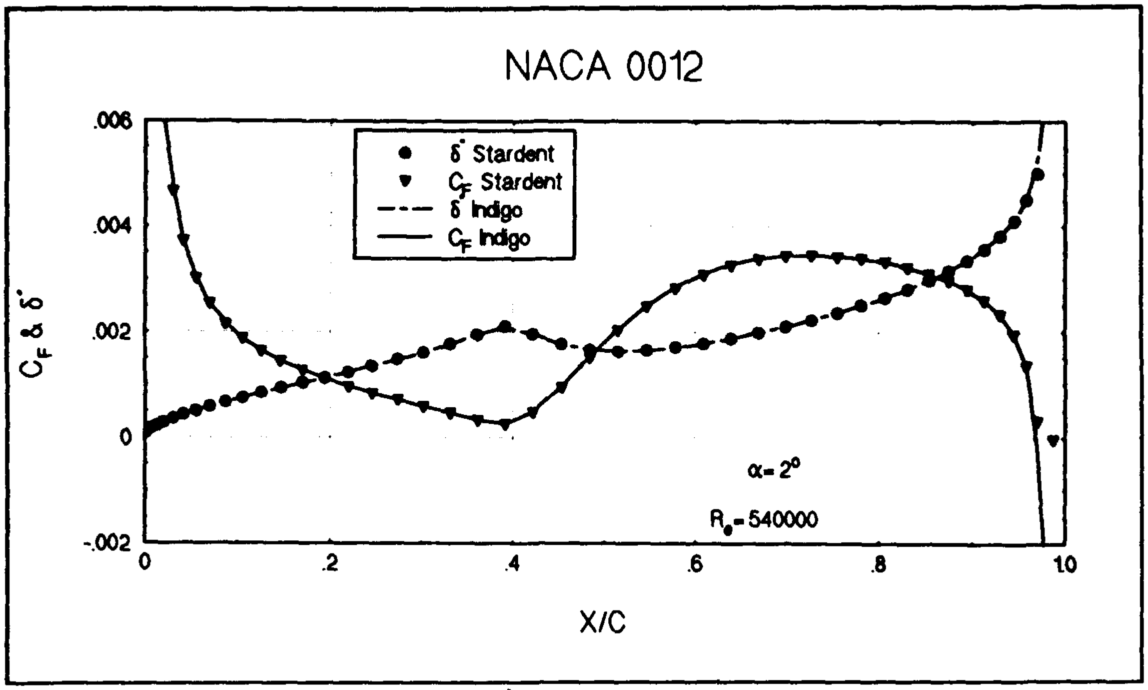


Figure 3.3

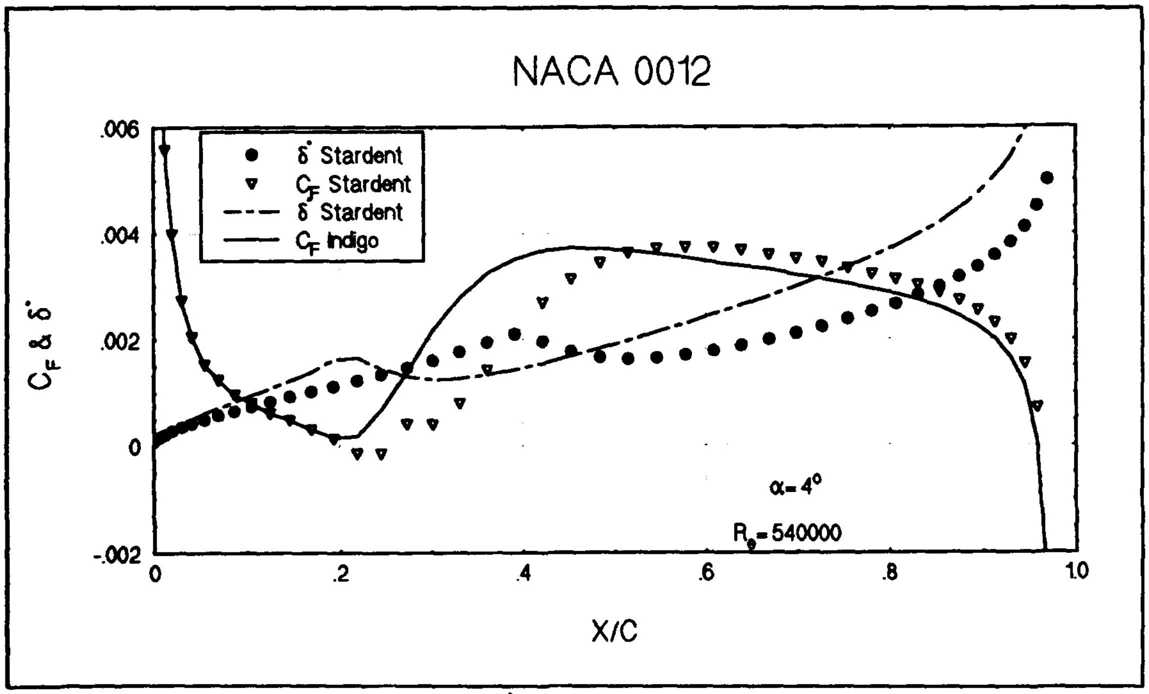


Figure 3.4

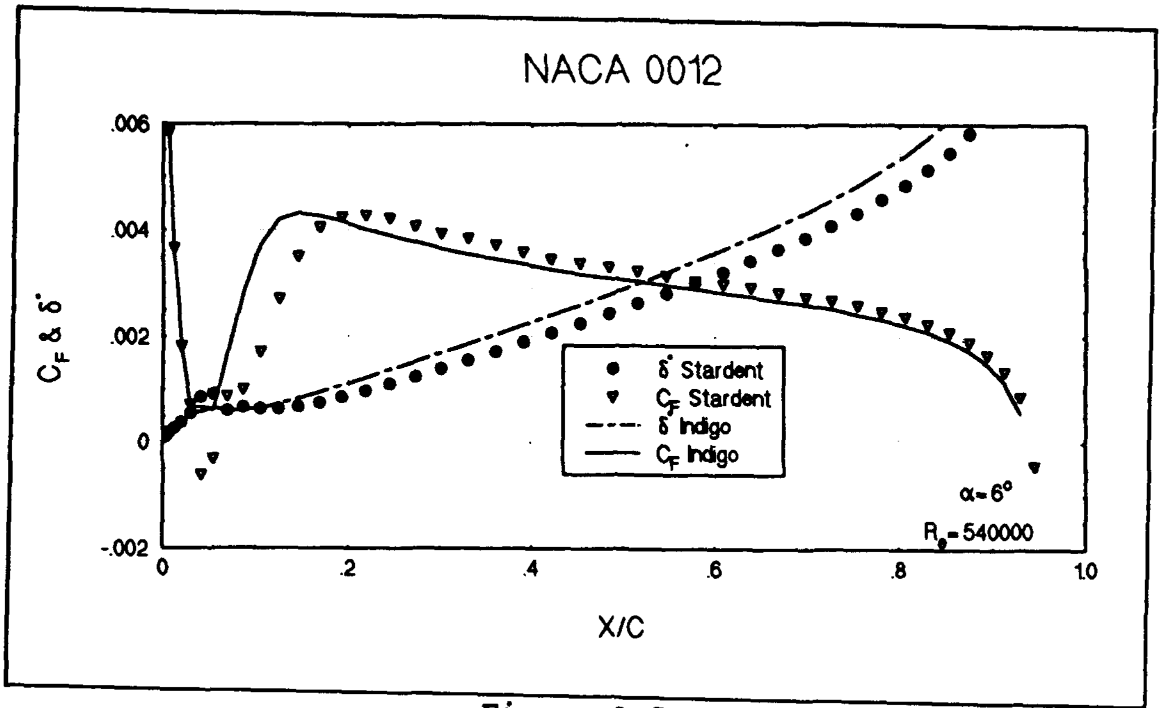


Figure 3.5

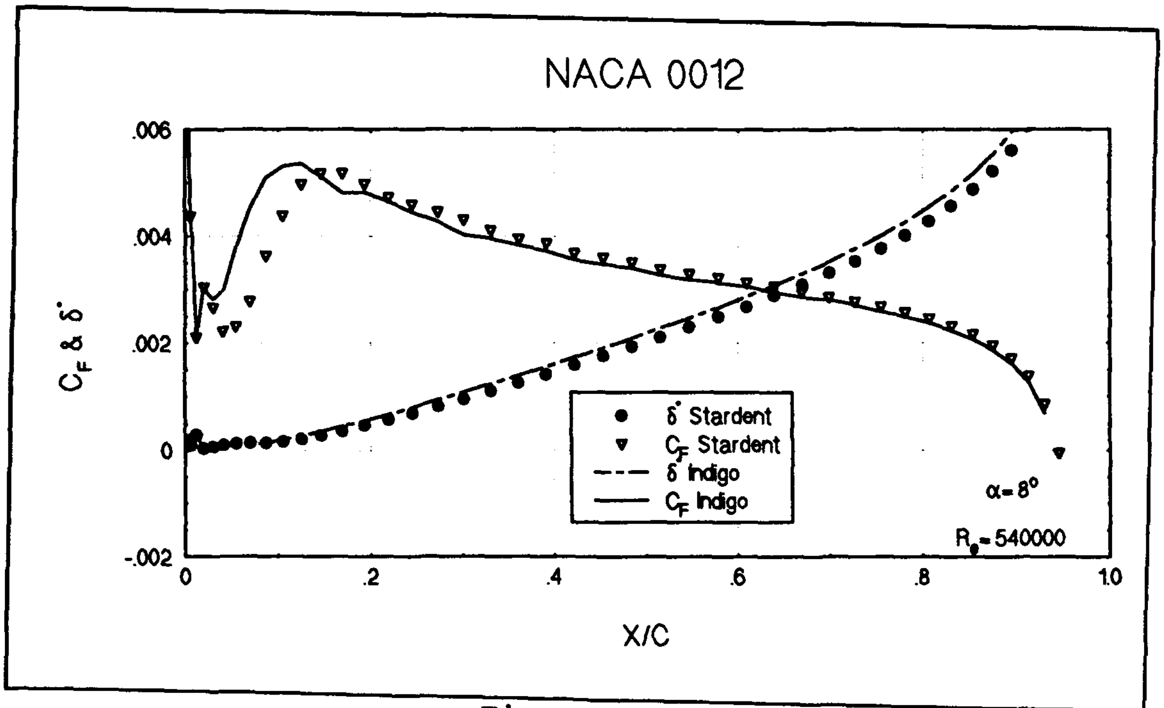
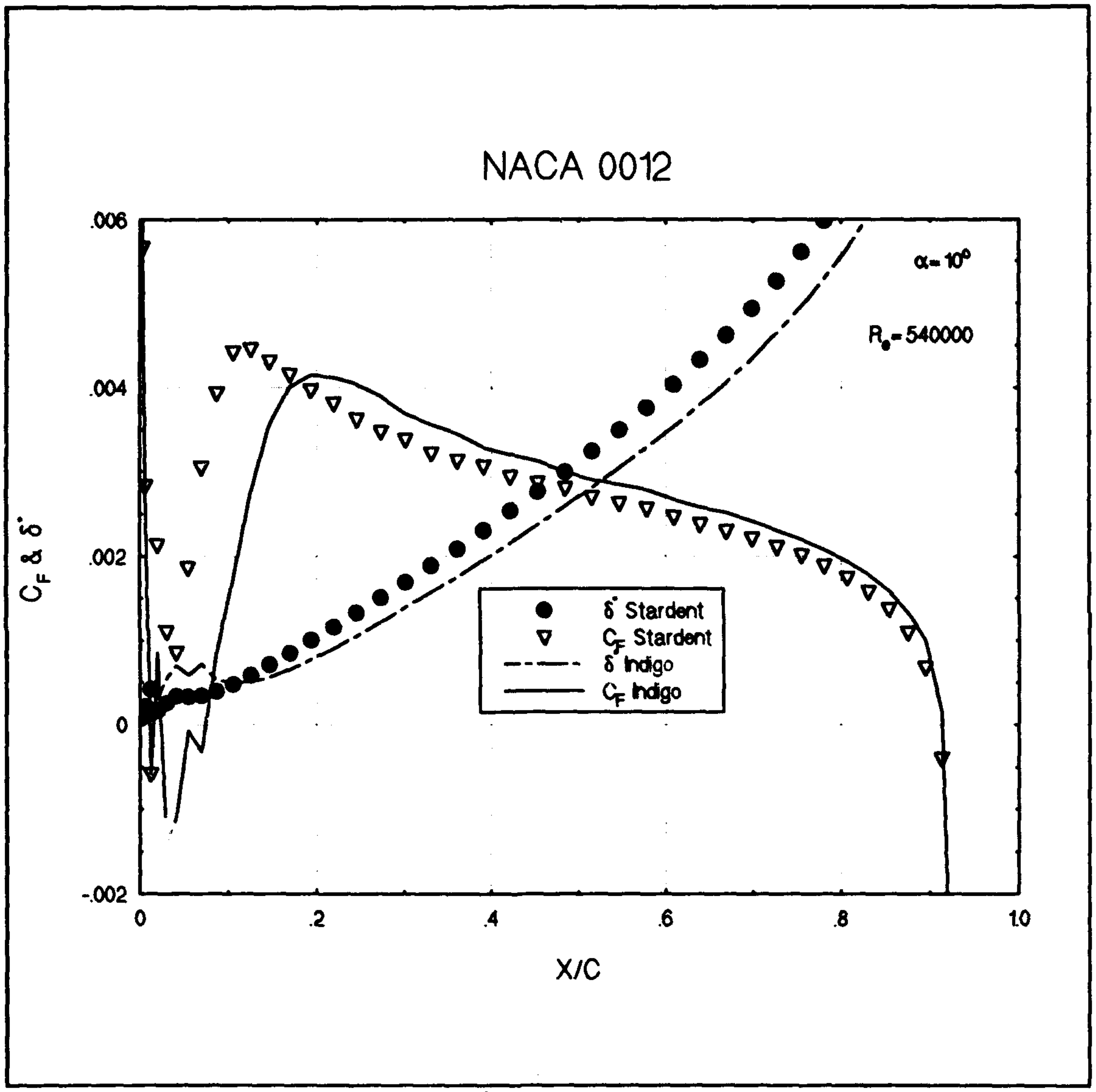


Figure 3.6



NACA 0012
$\alpha = 10^0$
$R_e = 540000$
δ^* Stardent
C_F Stardent
δ^* Indigo
C_F Indigo
C_F & δ^*
X/C
.006
.004
.002
0
-.002
0
.2
.4
.6
.8
1.0

Figure 3.7

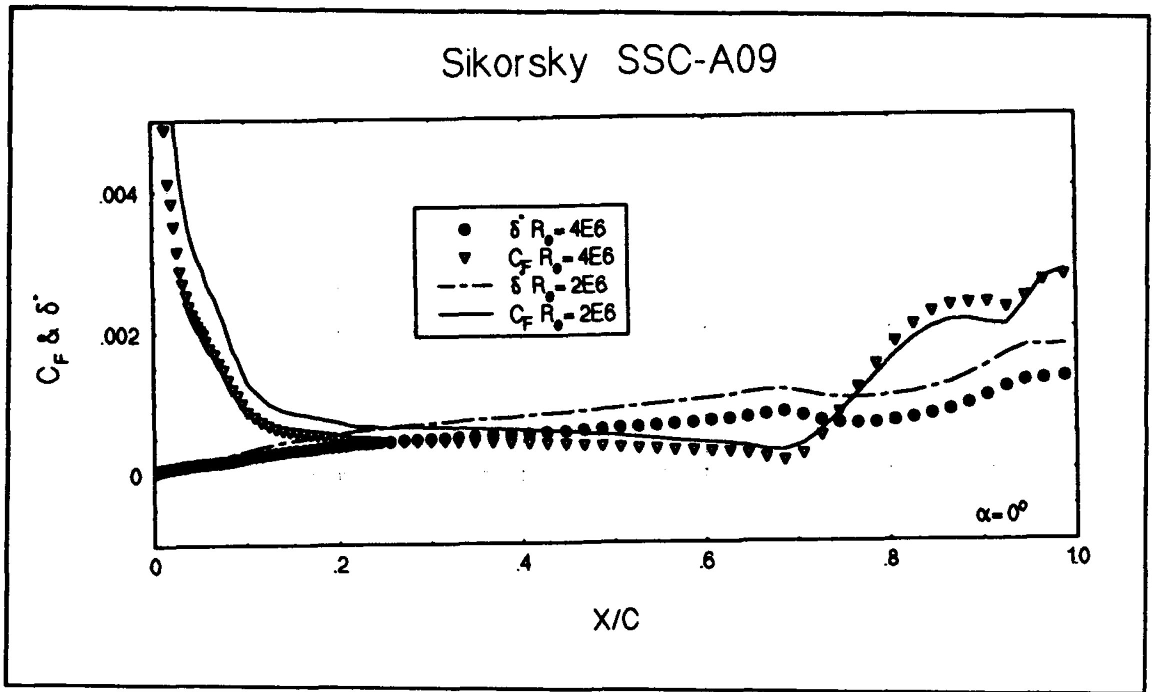


Figure 3.8

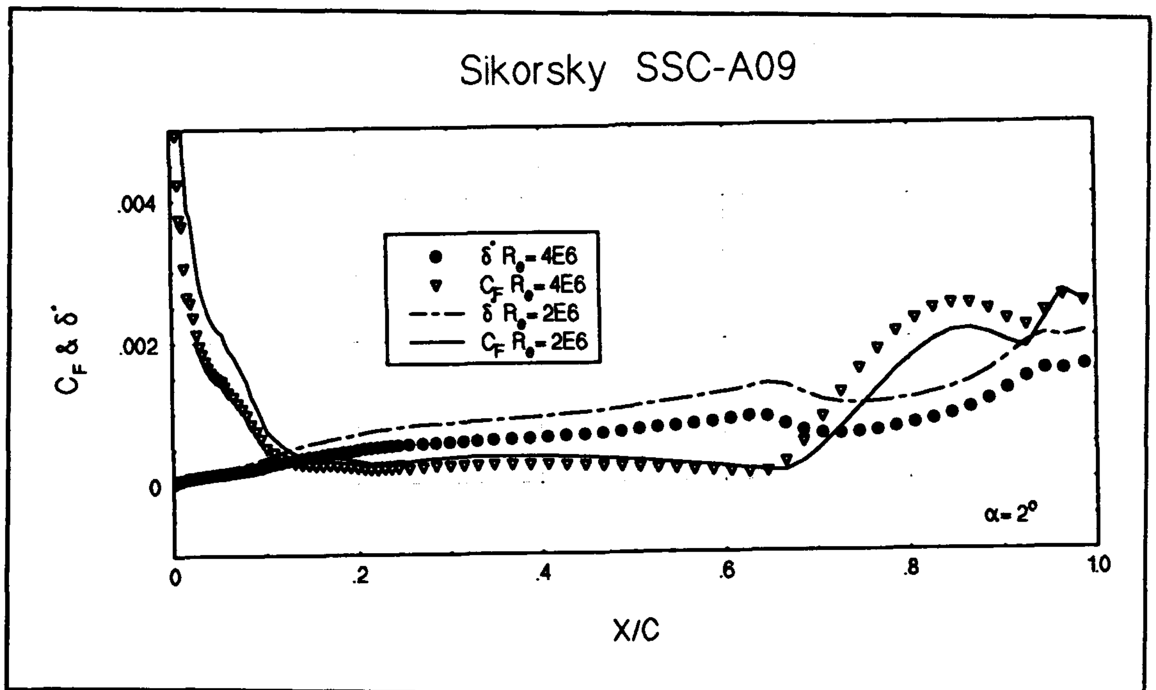


Figure 3.9

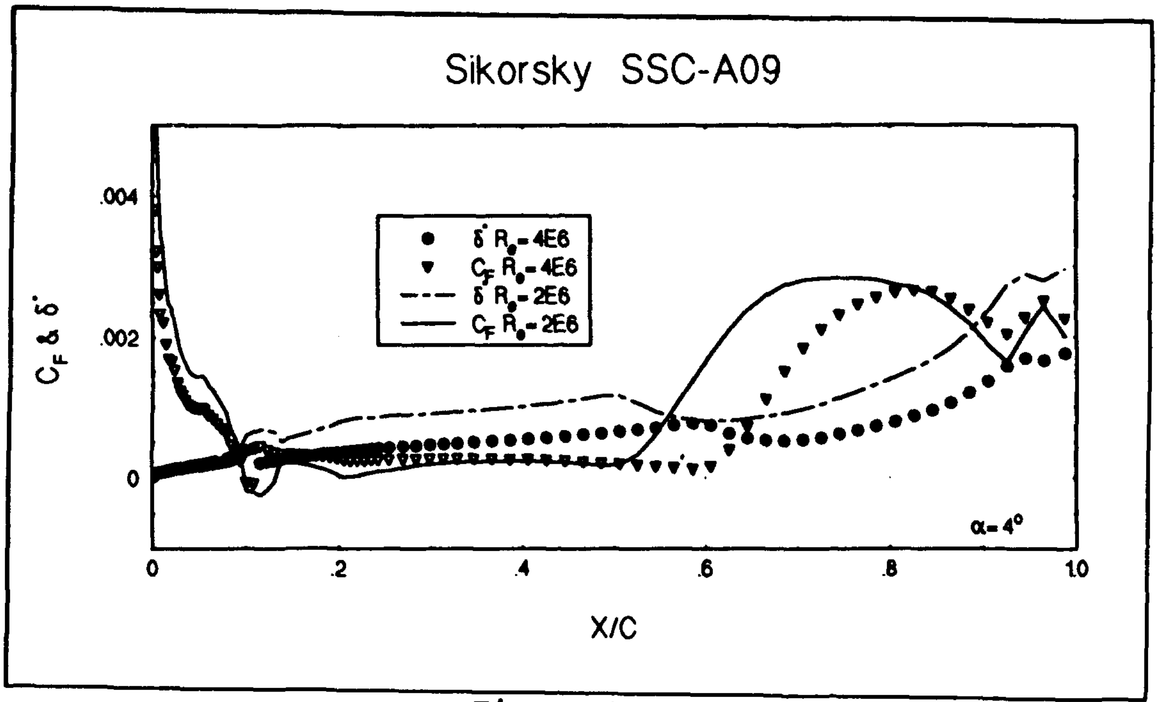


Figure 3.10

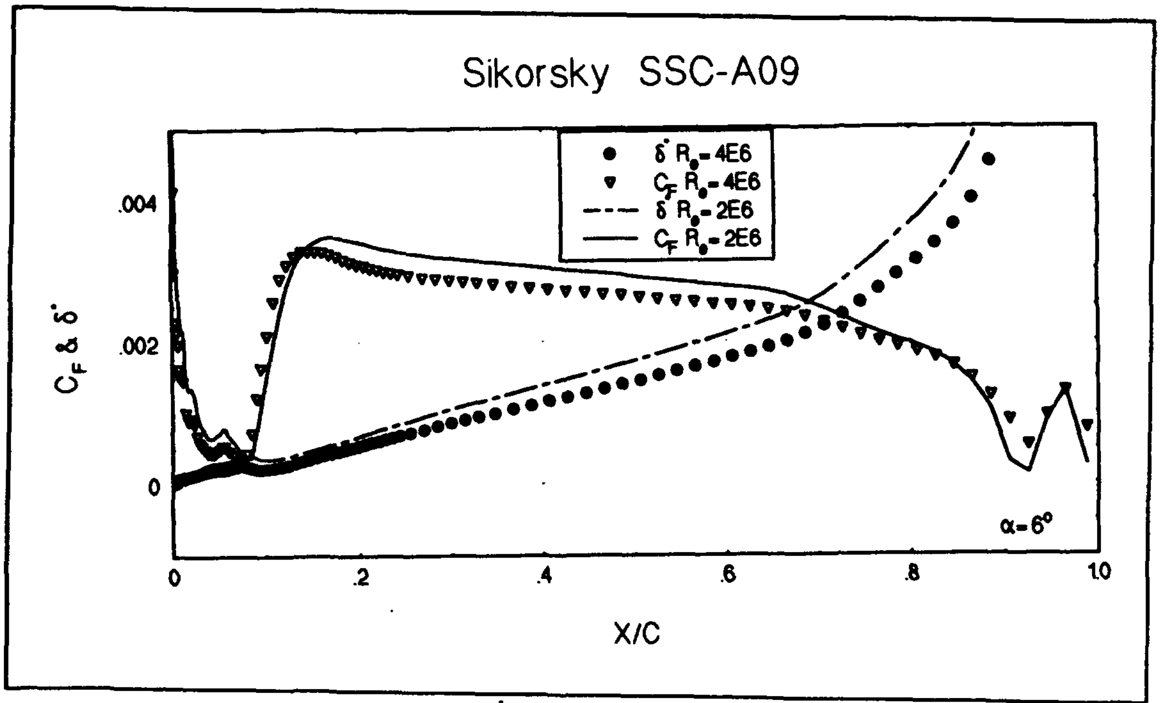


Figure 3.11

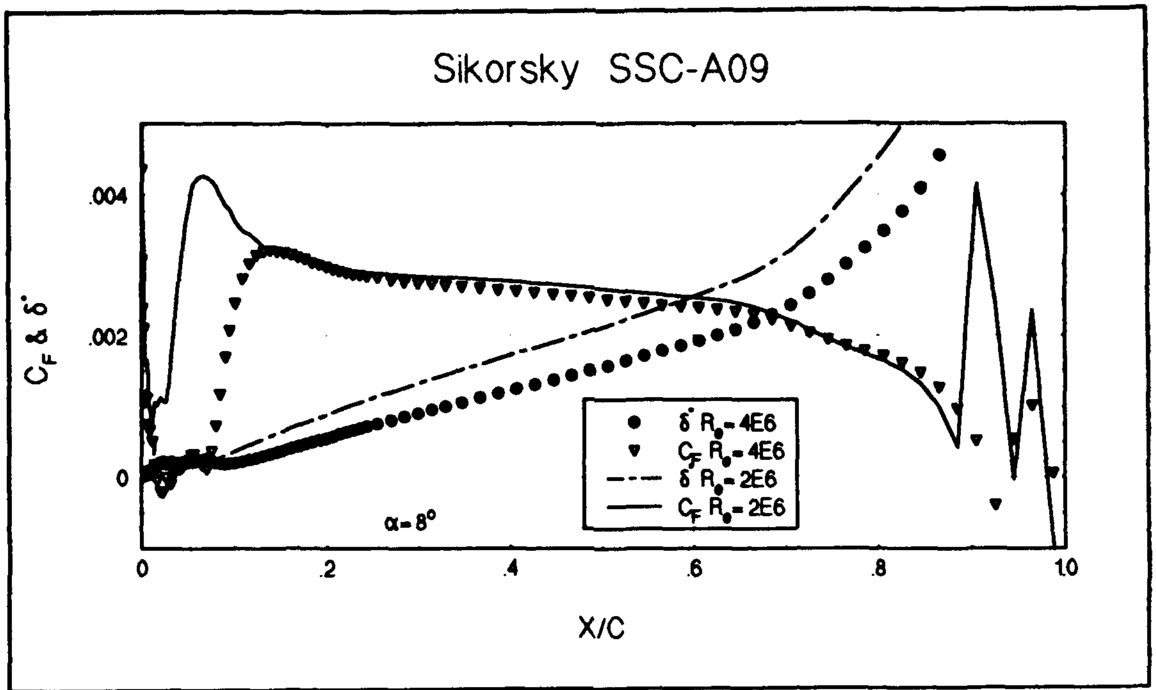


Figure 3.12

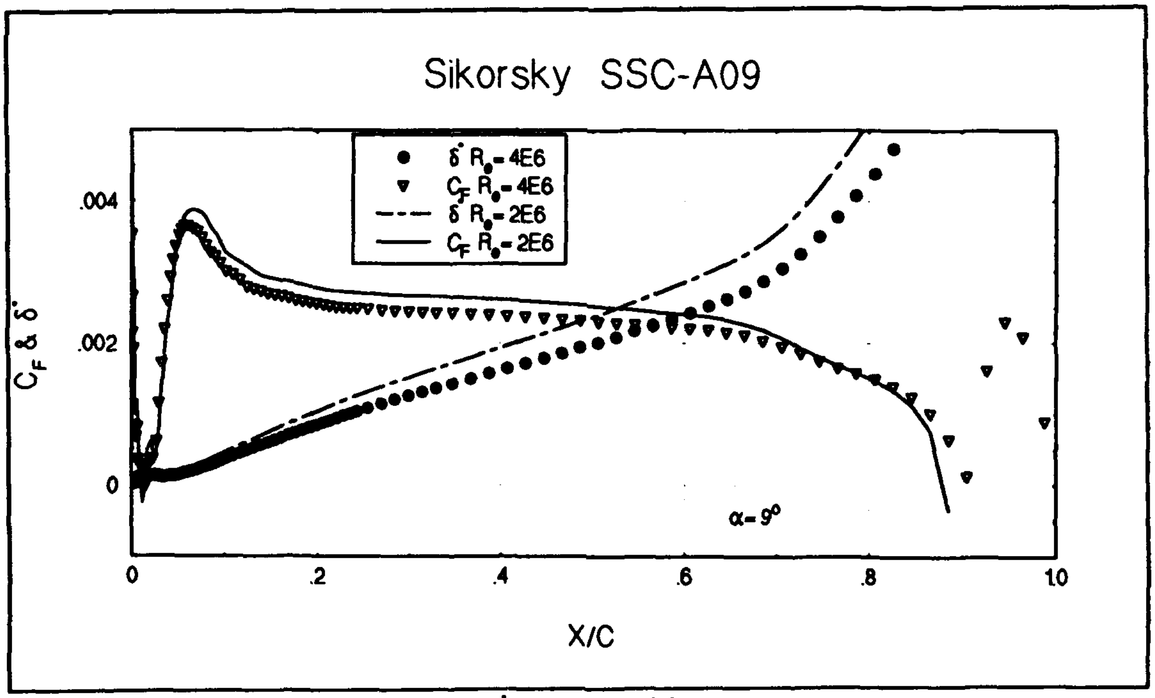


Figure 3.13

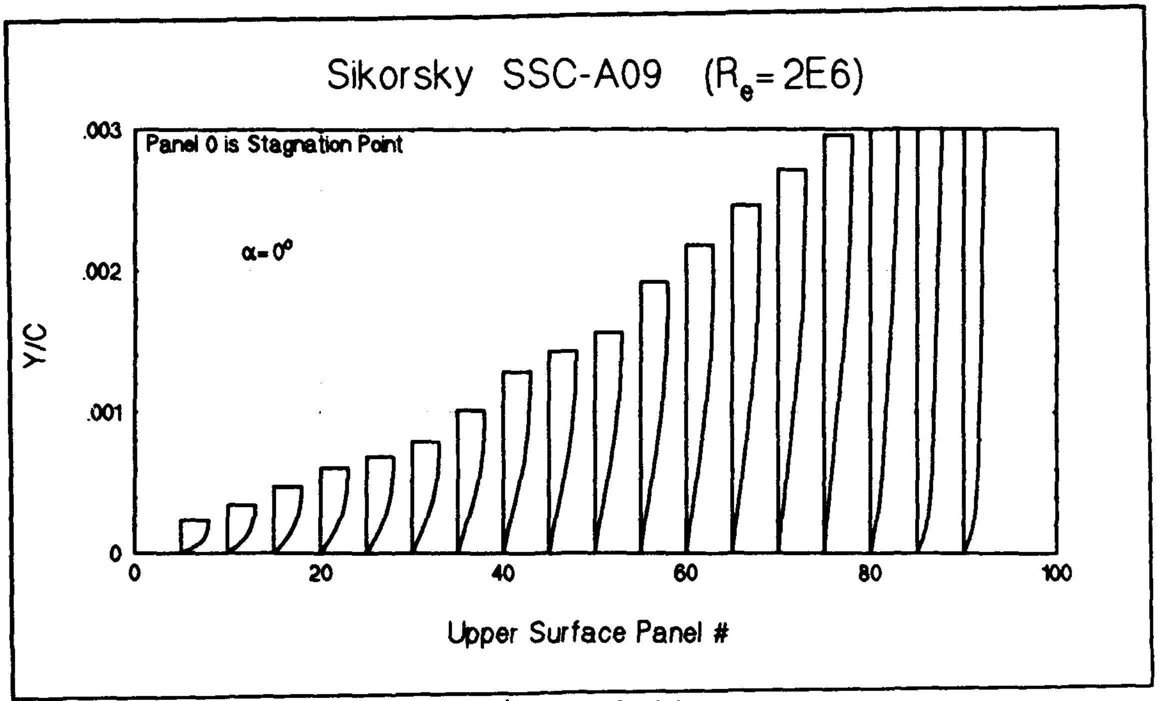


Figure 3.14

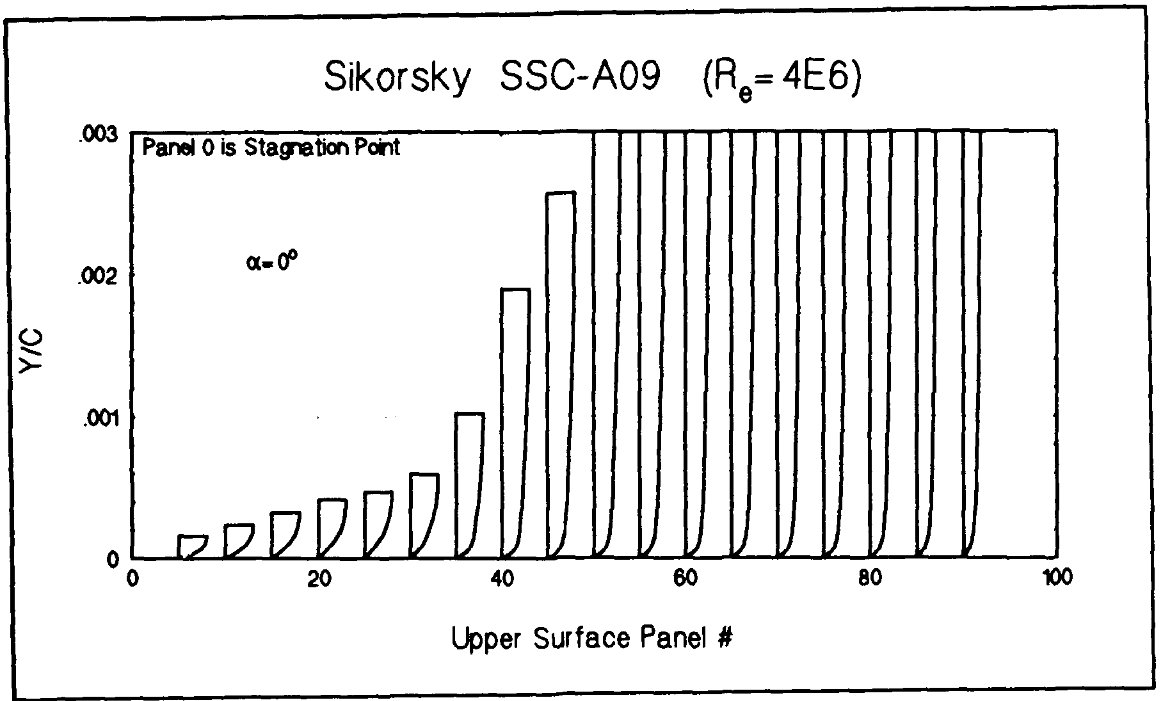


Figure 3.15

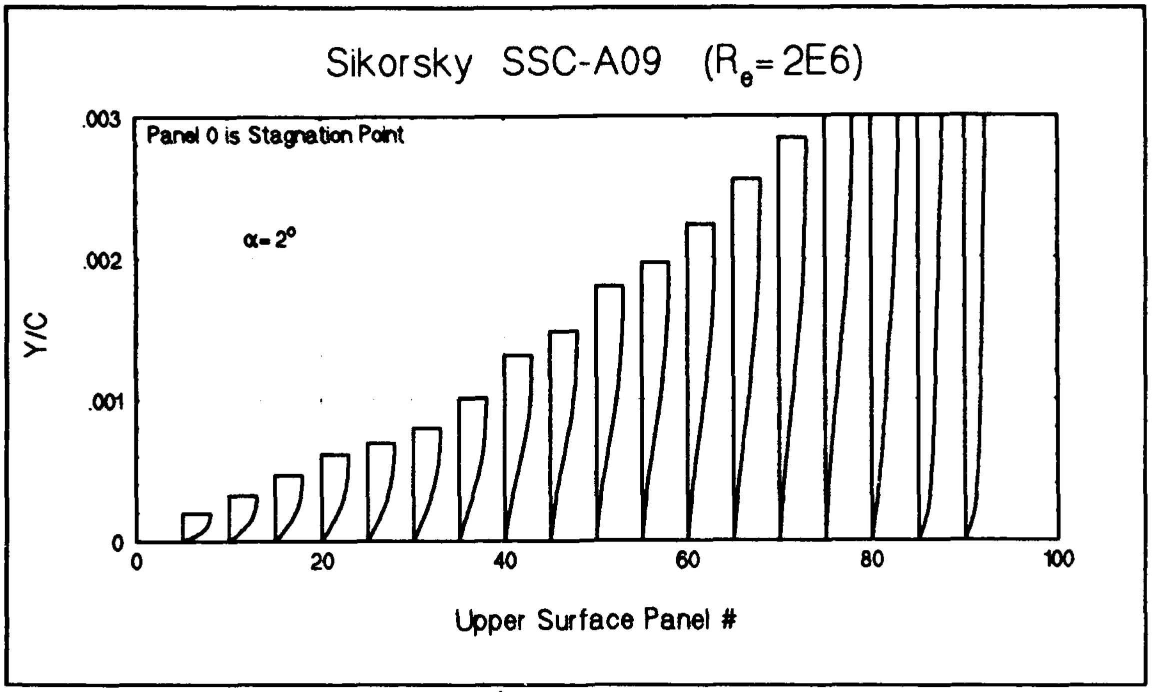


Figure 3.16

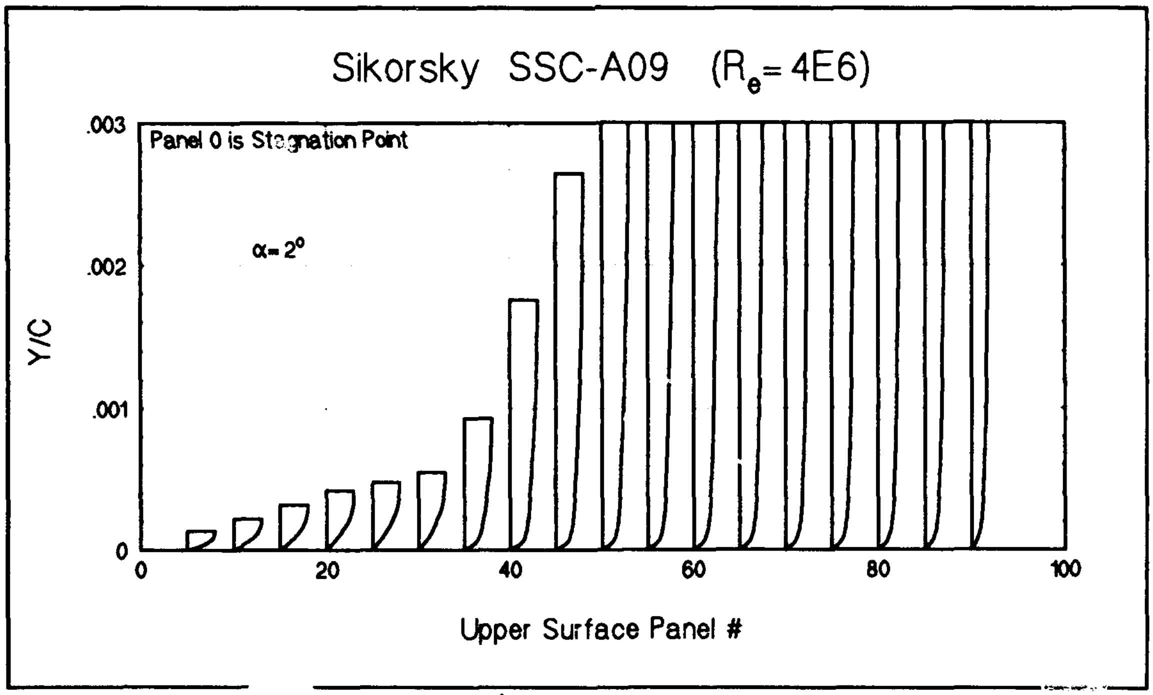


Figure 3.17

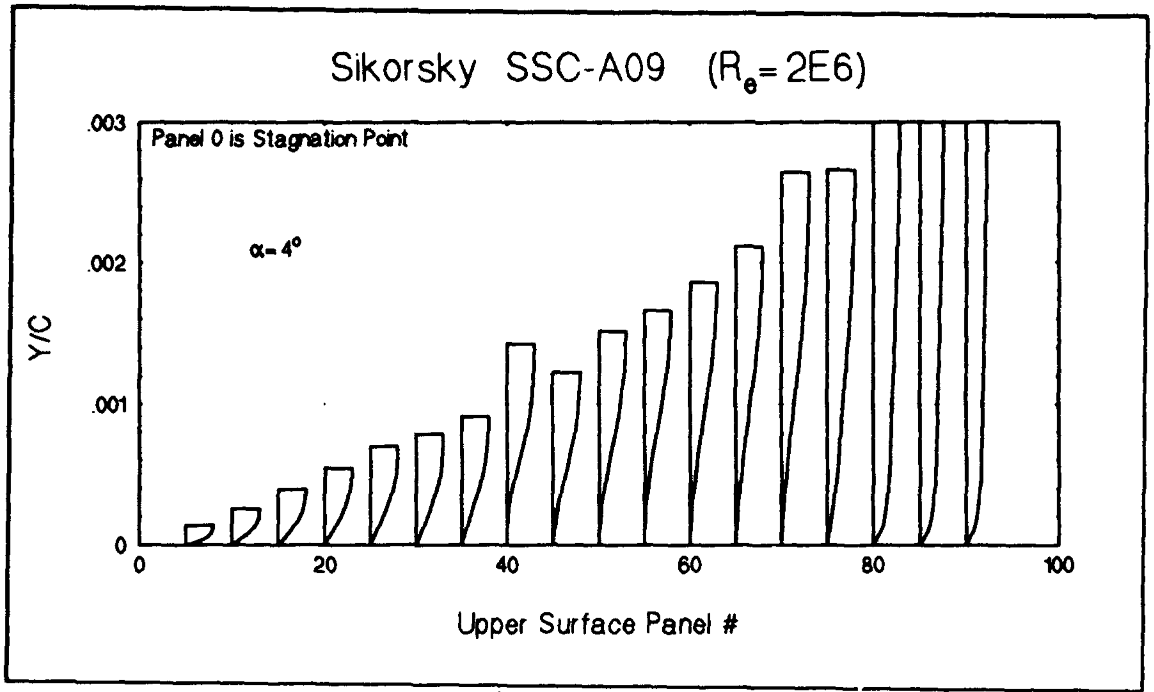


Figure 3.18

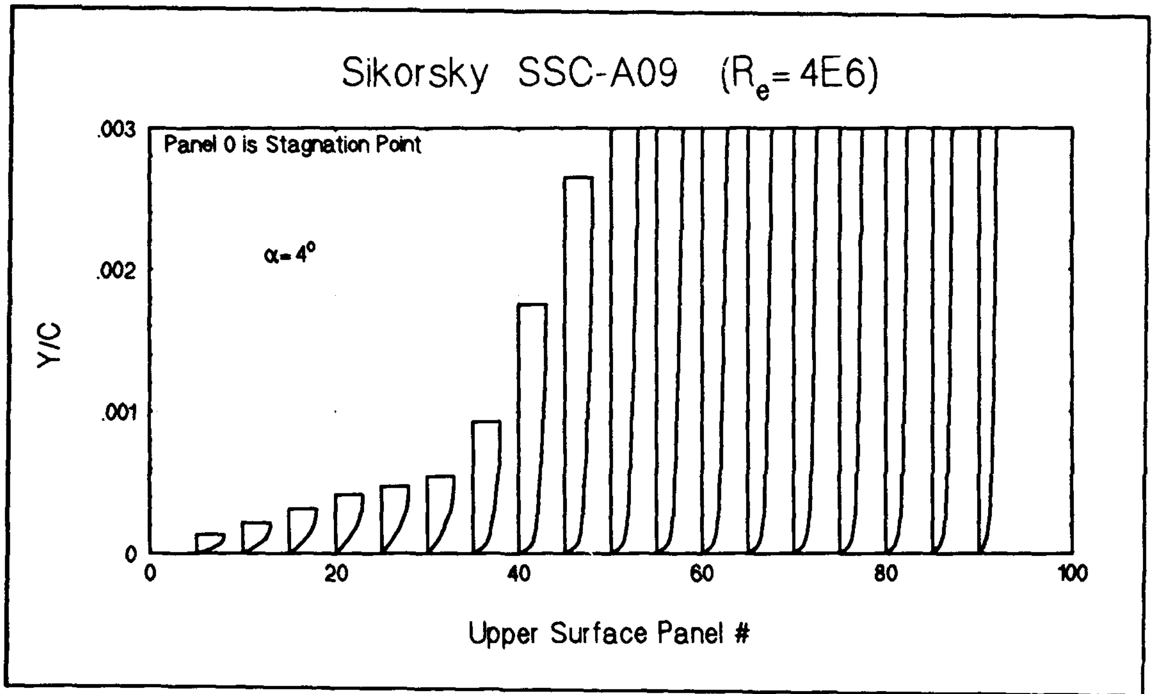


Figure 3.19

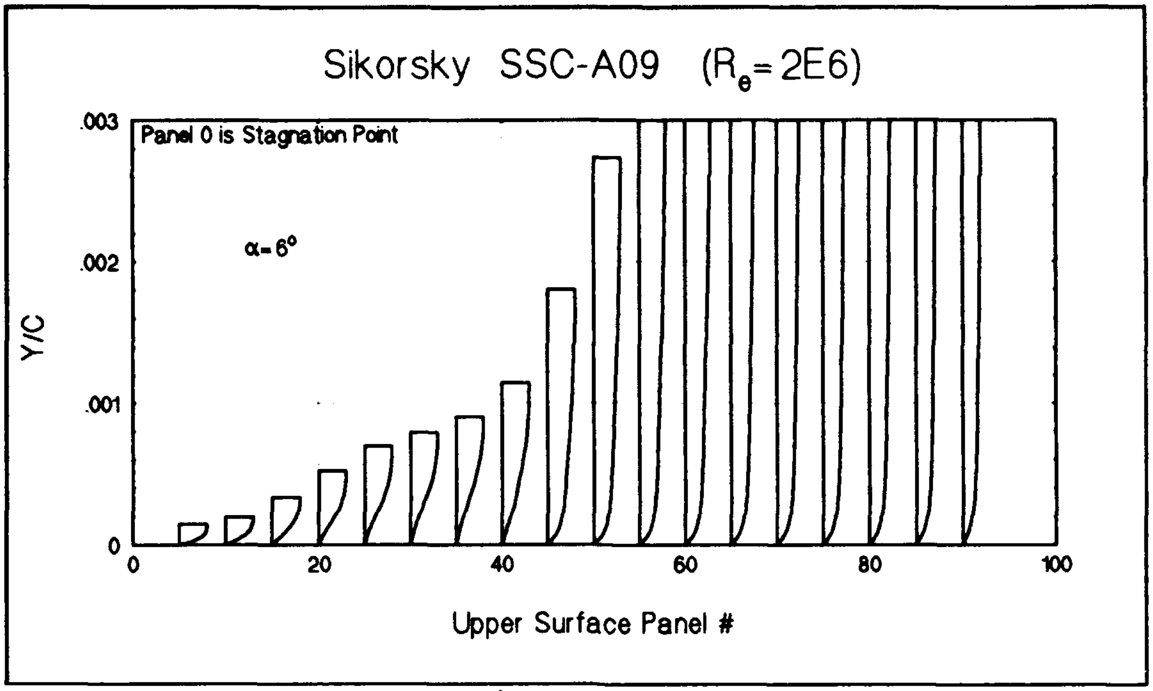


Figure 3.20

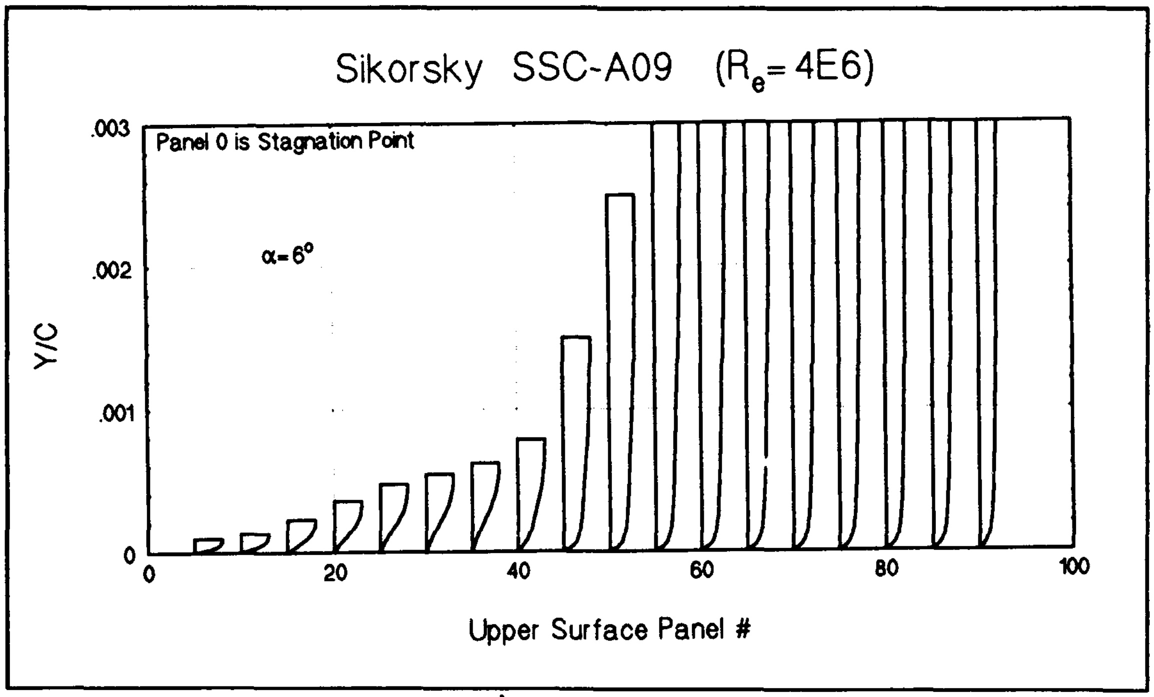


Figure 3.21

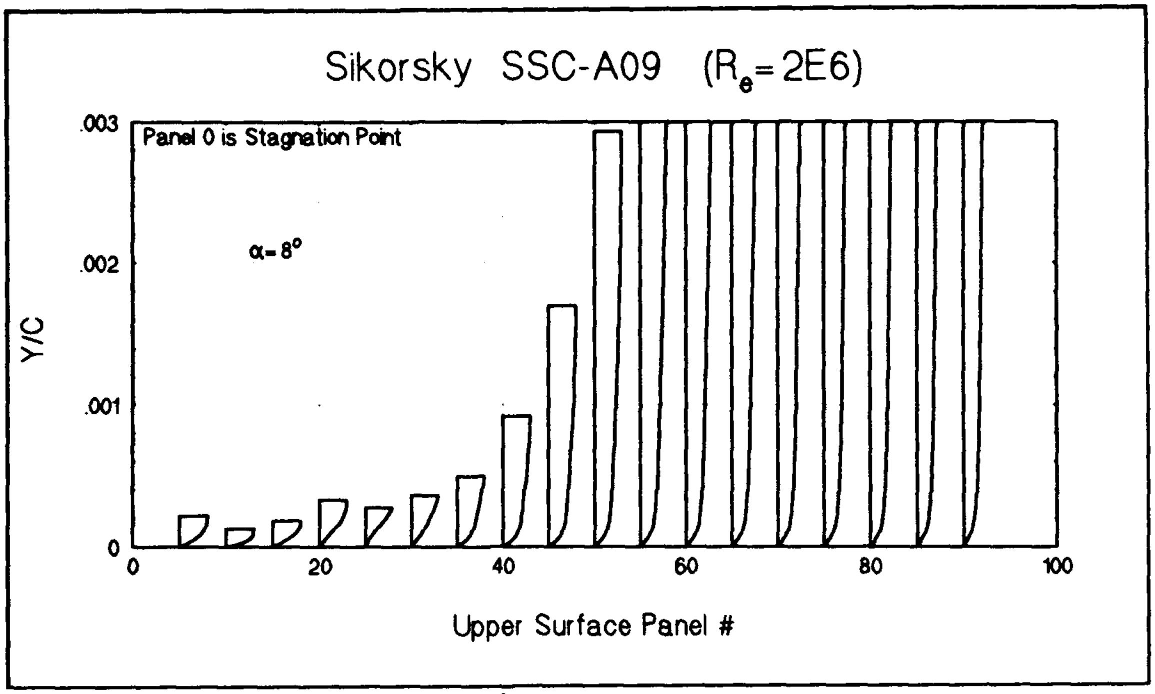


Figure 3.22

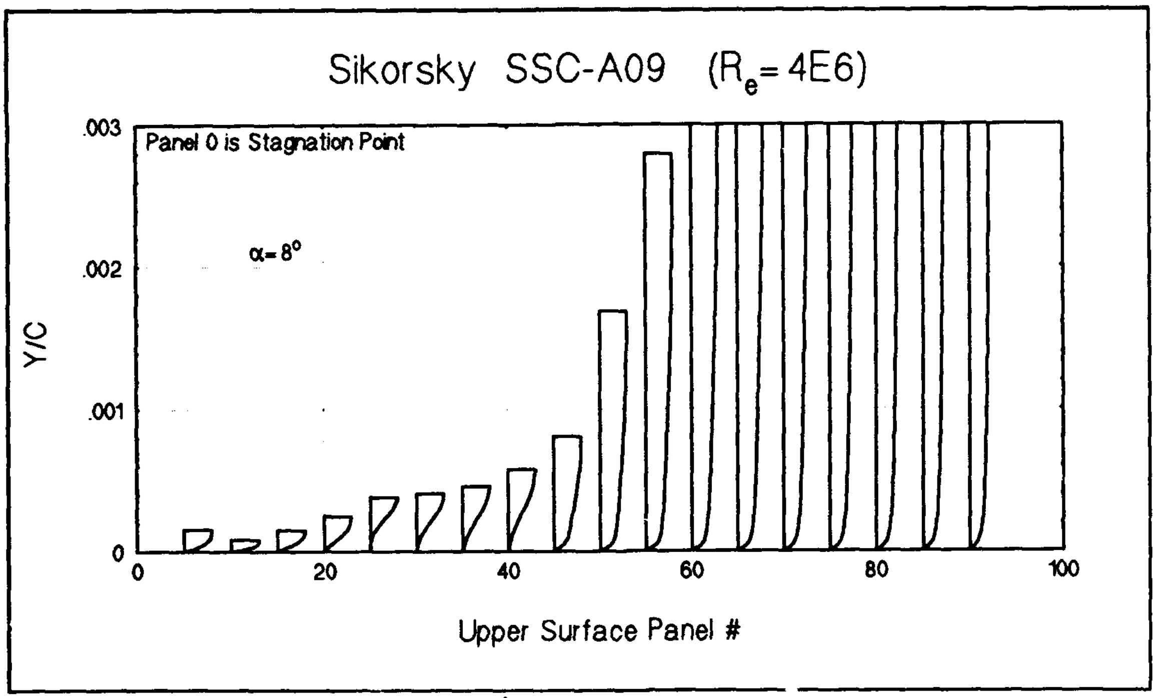


Figure 3.23

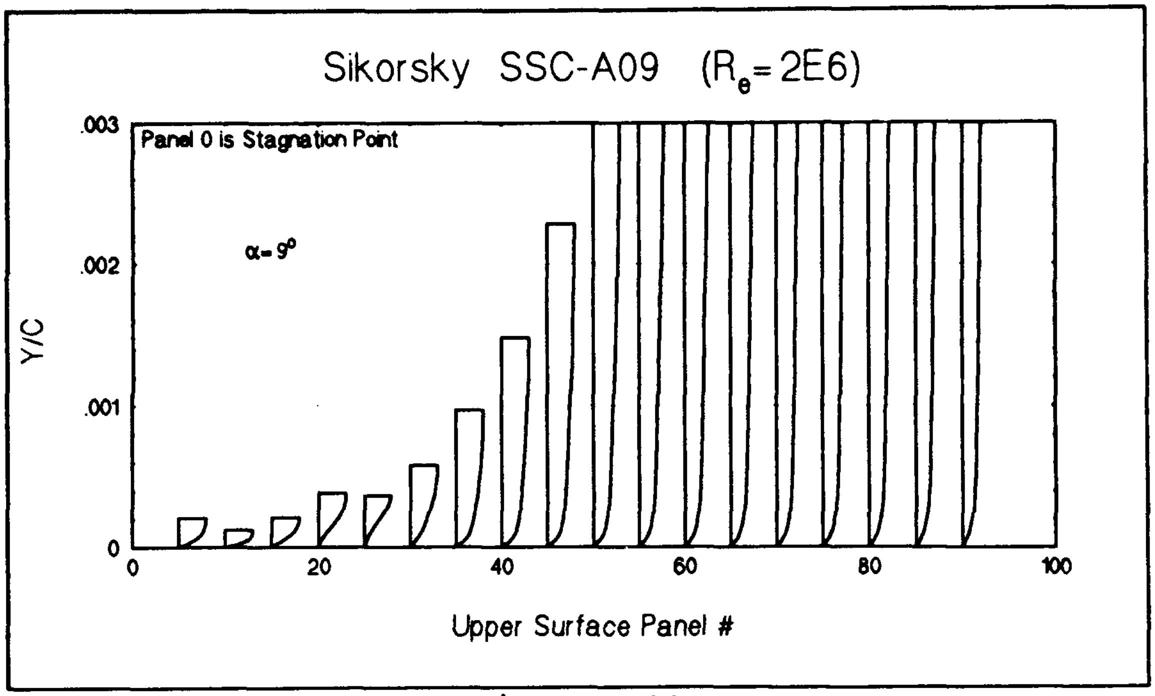


Figure 3.24

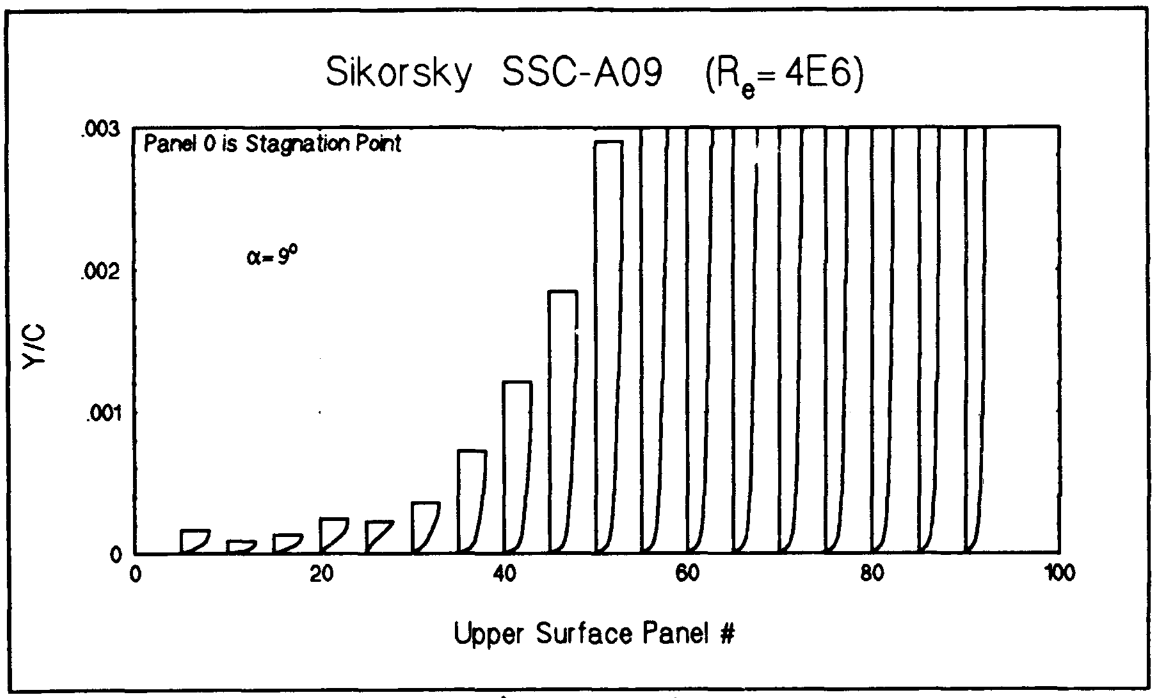


Figure 3.25

IV. UNSTEADY, LINEAR PANEL CODE

A. THEORY/BACKGROUND

The steady, linear panel code used in chapter II is adapted here to unsteady flow by building in a time dependency and modeling the vortex shedding process. Two further assumptions that are required are:

- The viscous flow effect must be negligible.
- The flow must stay attached on the airfoil surface.

Teng [Ref. 15] adapted such a formulation and is used here. This panel method was originally developed by Hess and Smith [Ref. 9] for steady flow. Its extension to unsteady motion was achieved by continuously shedding vorticity into a trailing wake using an interactive solution.

1. Flow Model

Complicating the unsteady flow solution are the now time dependent N+1 unknown singularity distributions (sources and vortices). These singularities are given a time index. As before, the source strengths are allowed to vary from panel to panel per time step; and the vorticity is a constant at each time step. The vortex shedding process can be defined through the basic definition of circulation, Equation 4.1, and the Helmholtz theorem of vortex continuity - that potential

flow total circulation must be preserved (Anderson [Ref. 2]). The airfoil perimeter is identified as p.

$$\Gamma_k = - \oint \{ V_k \cdot dS \} = \gamma_k \times p \qquad (4.1)$$

Therefore, circulation changes on the airfoil surface must be equal and opposite to the wake vorticity. Thus the shed vorticity model allows a mechanism for communication between time steps.

2. Boundary Conditions

The flow tangency and Kutta conditions are no longer linear. This requires an iterative numerical solution scheme. The flow tangency condition remains the same. The Kutta condition must now include the trailing edge panel's potential rate of change.

3. Solution Scheme

The disturbance potential, Equation 4.2, is complicated by adding in potential contributions from the shed vorticity panels and the wake core vortices. The disturbance potential must be calculated at every control point at each time step taking great care to only include velocity contributions due to disturbances. Complete modeling, numerical solution scheme, and disturbance potential details can be found in Teng [Ref. 15].

$$\Phi = \{ \phi_\infty + \phi_{source} + \phi_{vortex} + \phi_{shed\ vortex} + \phi_{core\ vorticity} \}$$

A complete program (UPOT.F) user's guide with input and output file examples and the source code are located in Appendix B. A few non-dimensional parameters must first be clarified. The Reduced Pitch Rate (A) used to classify ramp motion and the Reduced Frequency (k) used in sinusoidal motion can be based on full or half-chord. Program UPOT.F uses full chord, but the Lorber and Carta experimental results [Ref. 11] use half-chord as shown in equations 4.3 and 4.4.

$$A = \left\{ \frac{\dot{\alpha}\ c}{2\ U} \right\} \tag{4.3}$$

$$k = \left\{ \frac{\omega\ c}{2\ U} \right\}, \quad \omega = \mathit{Osillation\ Frequency} \tag{4.4}$$

B. THE NACA 0012 AIRFOIL

Calculated force and moment coefficients as a function of angle-of-attack during a 0.005 Reduced Pitch Rate ramp motion are displayed in Figures 4.1 through 4.3. Steady state results from chapter two are also displayed for comparison. Very little noticeable difference is noted in lift or moment coefficient.

Results for a sinusoidal motion with a Reduced Frequency of 0.025 and a pitch magnitude of 12° are shown in Figures 4.4 through 4.6, again with steady state results previously obtained. Results for a sinusoidal motion with a Reduced Frequency of 0.05 and a pitch magnitude of 20° are shown in

Figures 4.7 through 4.9. Here, lift is augmented throughout the down cycle and lost during the up cycle, contrary to what would be expected in experiment. Similar results are shown for the drag and the moment coefficients.

C. THE SIKORSKY SSC-A09 AIRFOIL

To validate the Lorber and Carta experimental data, a ramp motion with a Reduced Pitch Rate of 0.005 and sinusoidal motions with Reduced Frequencies of 0.025 and 0.05 were completed. UPOT.F unsteady, PANEL.F steady, and Lorber and Carta experimental data are presented in Figures 4.10 through 4.18.

1. Ramp Motion, A=0.005 (0° to 20°)

The calculated steady state and unsteady lift coefficient varied little from the experimental results throughout the linear range (0° to 14°). Once nonlinear, viscous effects dominated the real flowfield, calculated results diverged as expected. Steady and unsteady calculated drag results follow the general direction of experimentally measured values but differ widely in magnitude due to the basic inviscid flow assumptions. Pitching-moment coefficient agrees well with experimental results through approximately 11° α.

2. Sinusoidal Motion

a. $\alpha(t) = 6 - 6\cos(\omega t)$, $k=0.025$, $0^\circ \rightarrow 12^\circ \rightarrow 0^\circ$

Computed unsteady, inviscid, incompressible results indicate a net loss of lift on the up cycle and augmented lift on the down cycle when compared to PANEL.F computed steady state values. UPOT.F overpredicts lift on both up and down cycles when compared to experiment. Pitching-moment coefficient results show a similar overall trend but are also displaced by a constant magnitude from experimentally measured results.

b. $\alpha(t) = 10 - 10\cos(\omega t)$, $k=0.05$, $0^\circ \rightarrow 20^\circ \rightarrow 0^\circ$

Similar results were obtained when the oscillation magnitude and Reduced Pitch rate were increased. Computed unsteady, inviscid, incompressible results indicate a net loss of lift on the up cycle and augmented lift on the down cycle when compared to PANEL.F computed steady state values. No reasonable correlation could be drawn from the drag results due to the code's inviscid flow assumption. The general pitching-moment coefficient trend is similar within the linear region, but varies wildly within the nonlinear region.

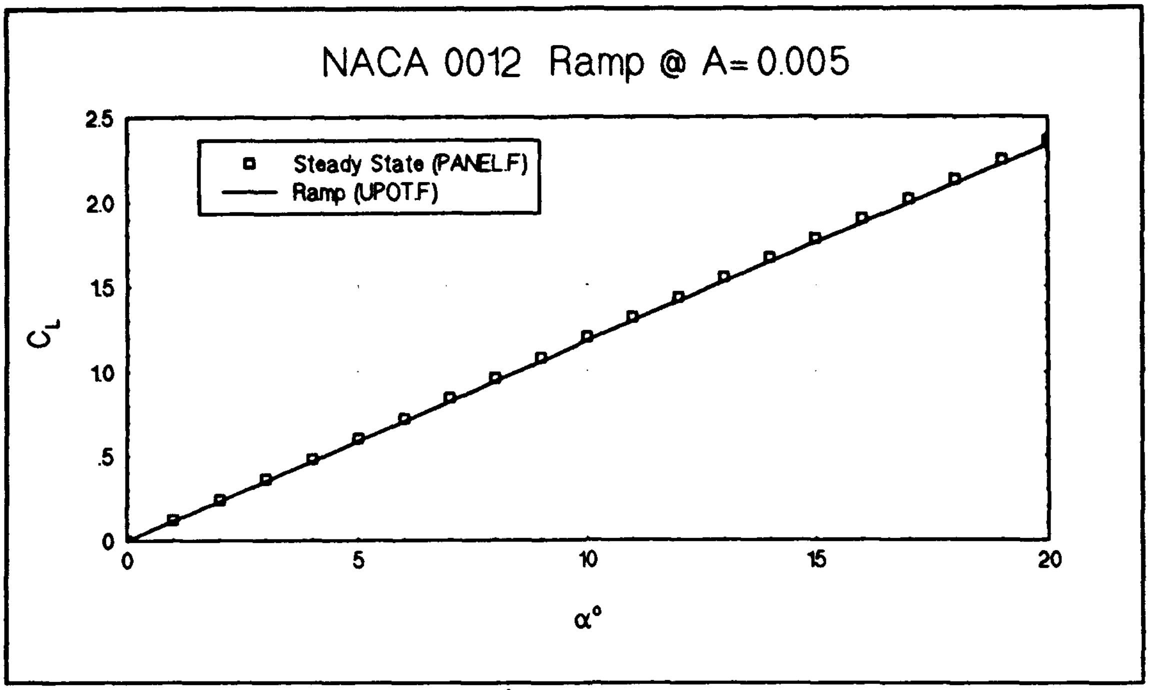


Figure 4.1

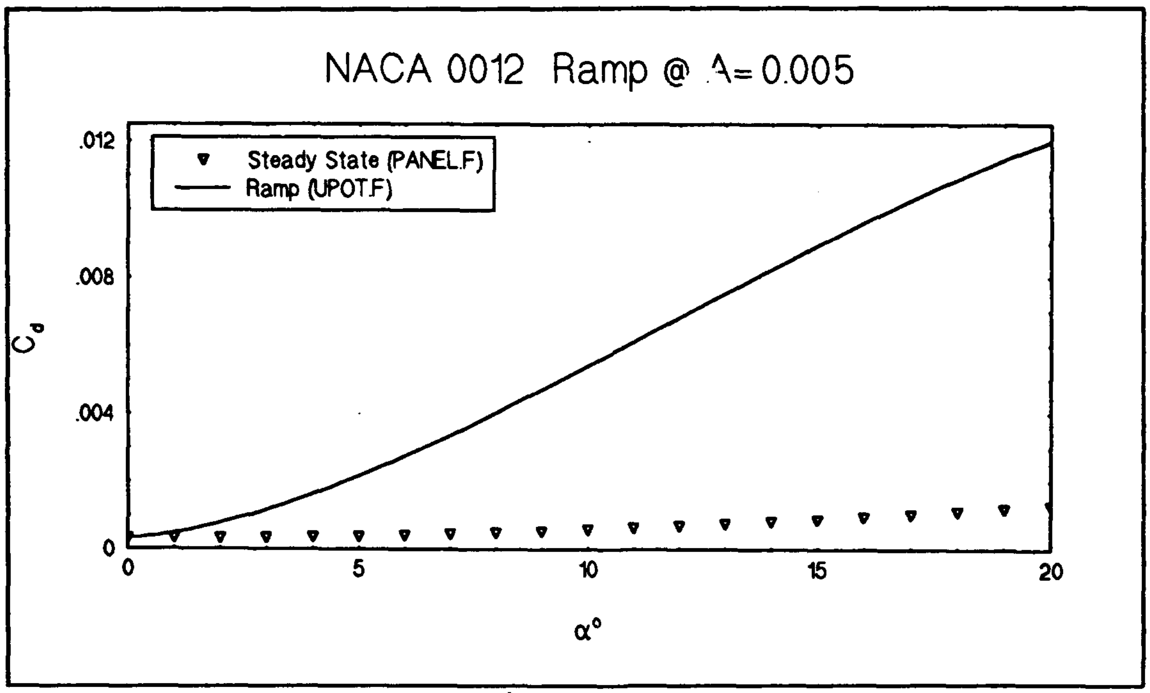


Figure 4.2

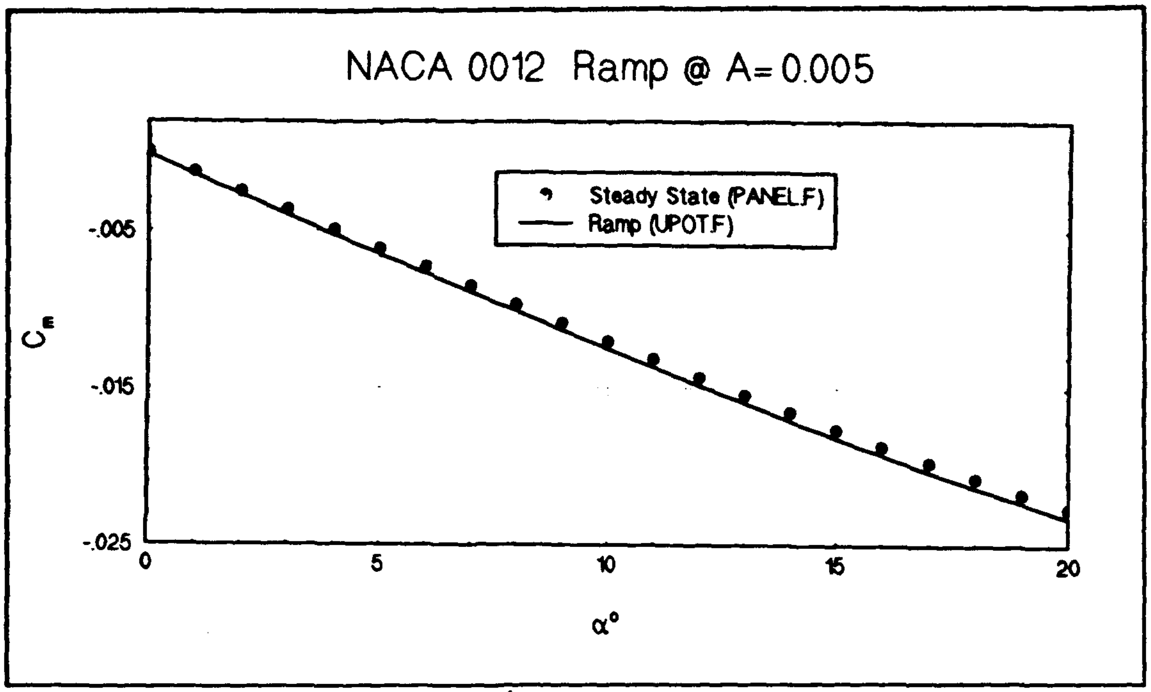


Figure 4.3

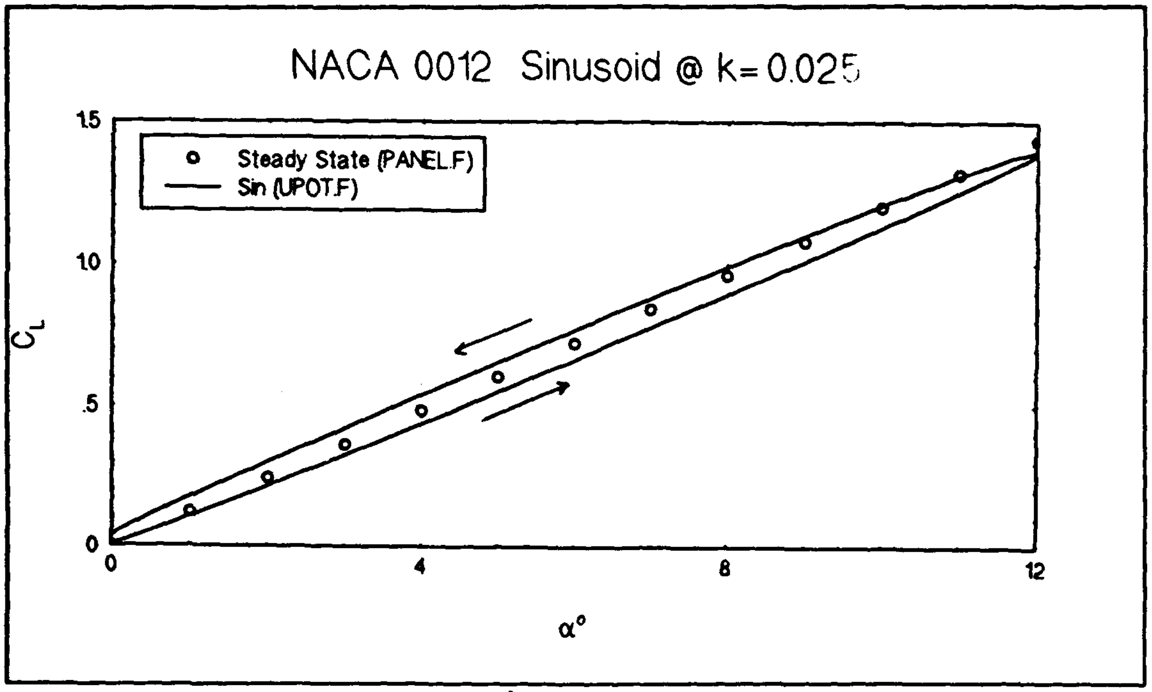


Figure 4.4

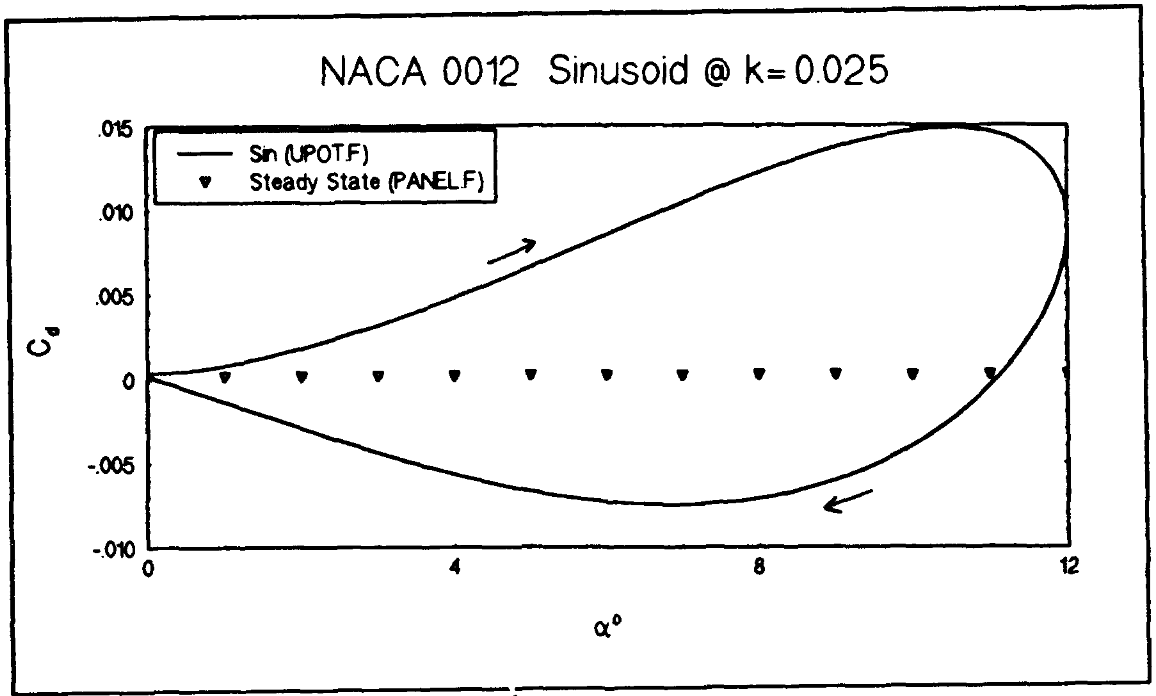


Figure 4.5

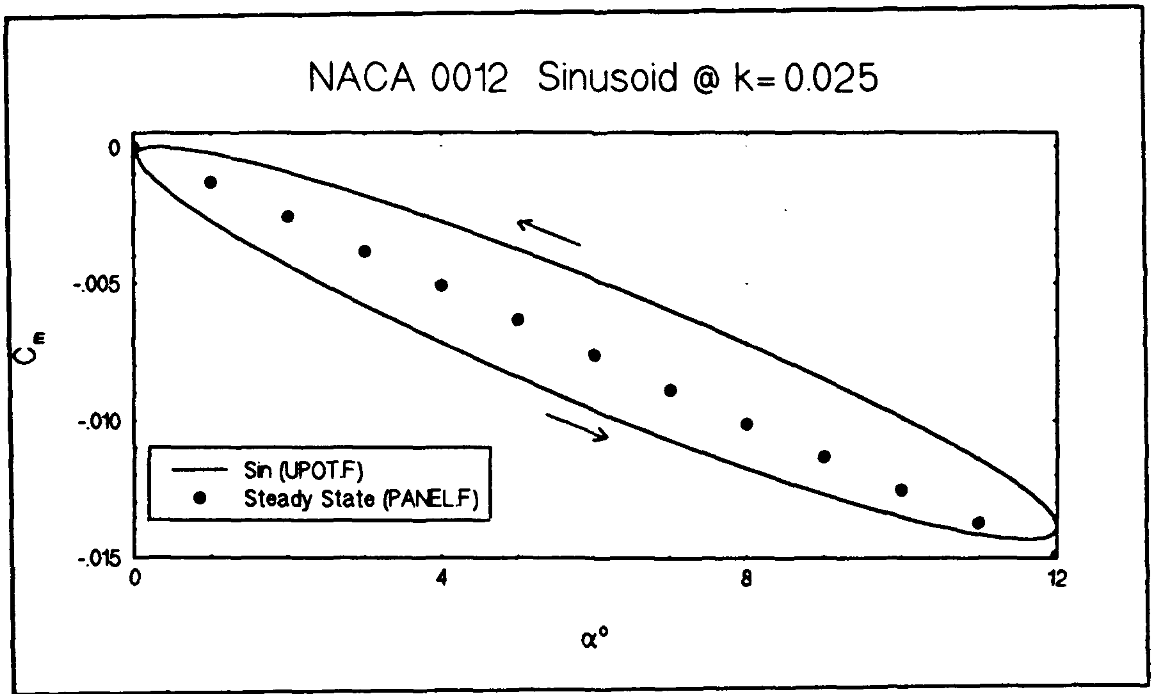


Figure 4.6

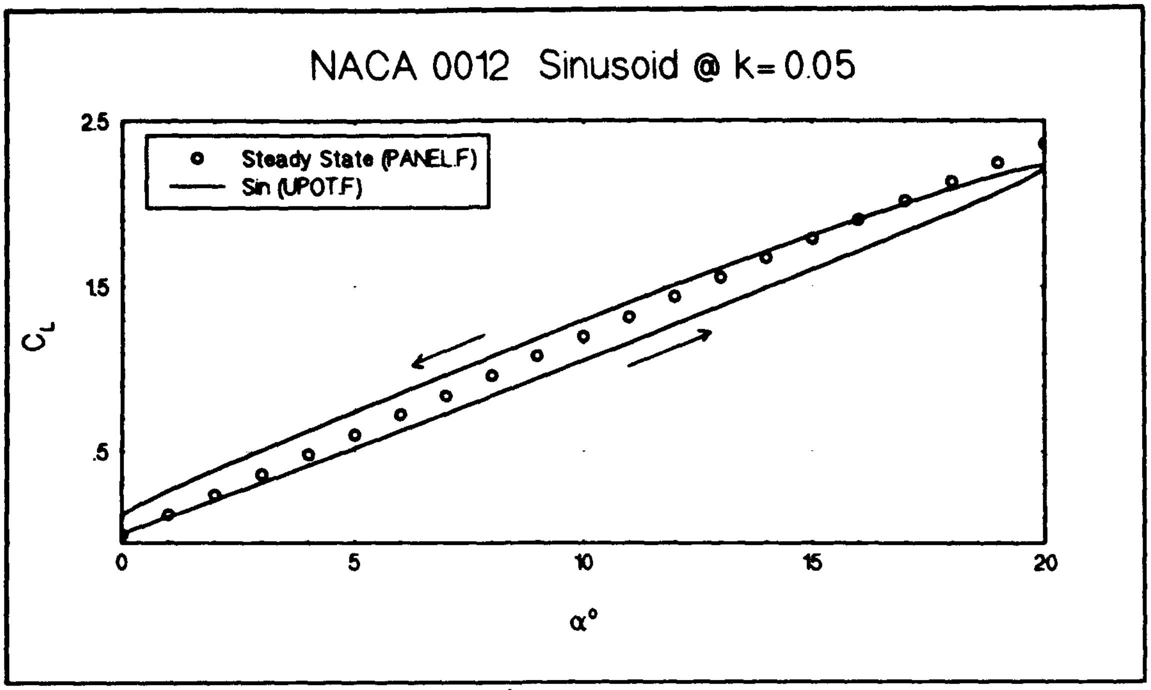


Figure 4.7

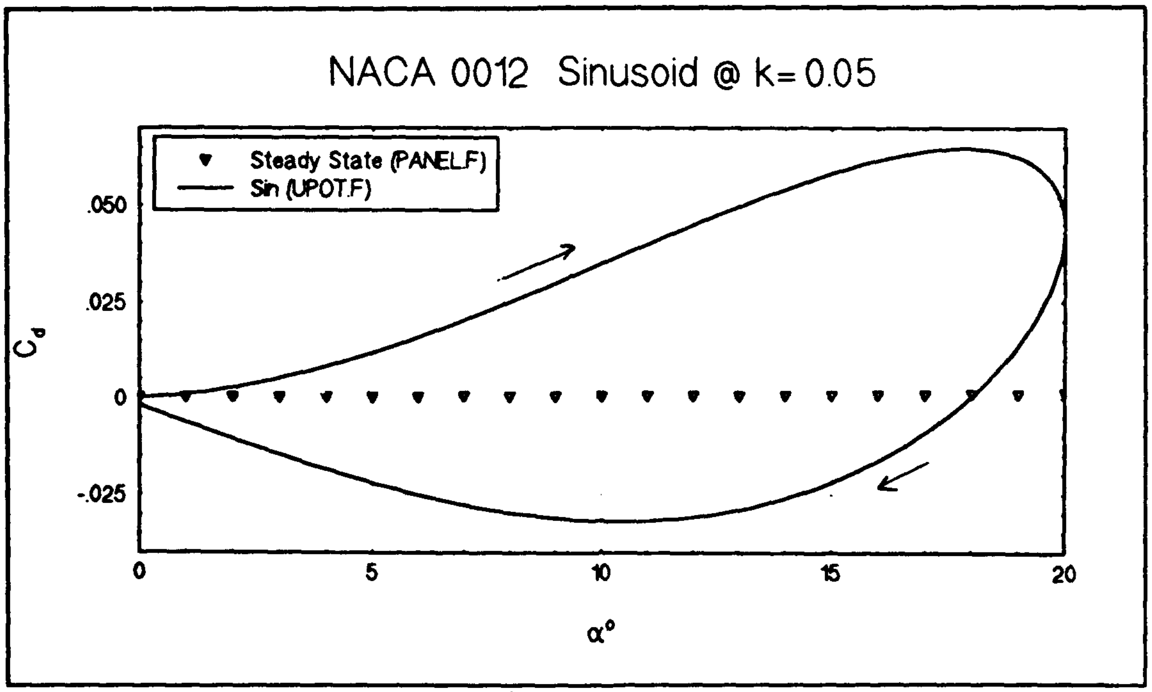


Figure 4.8

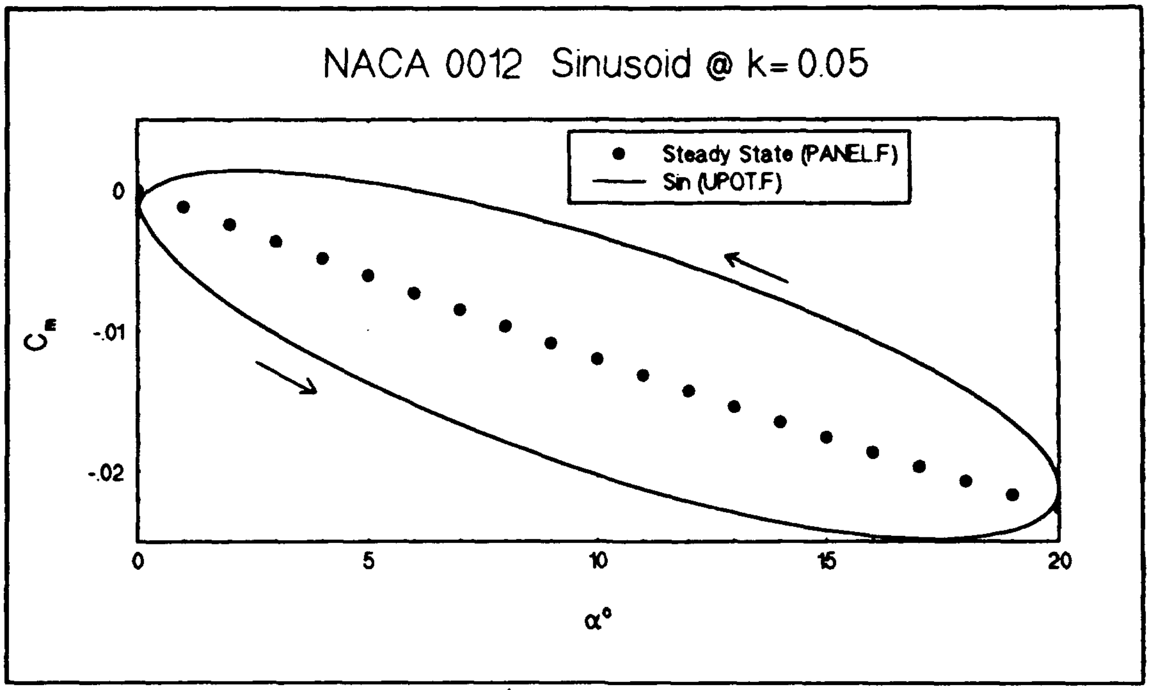


Figure 4.9

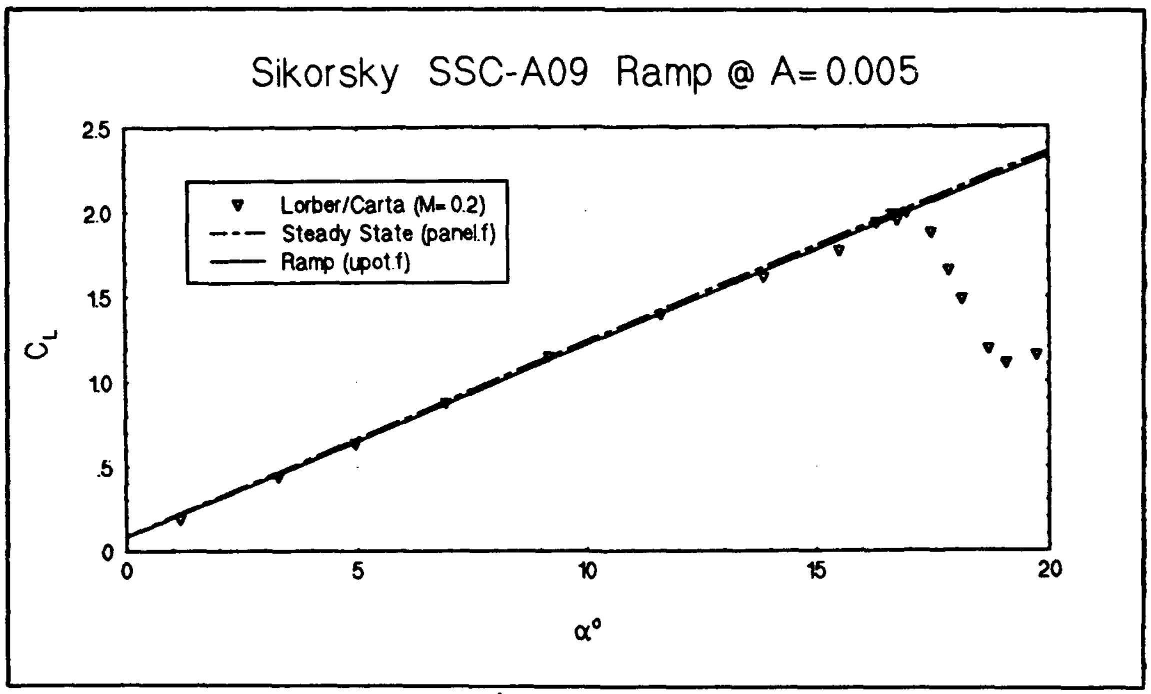


Figure 4.10

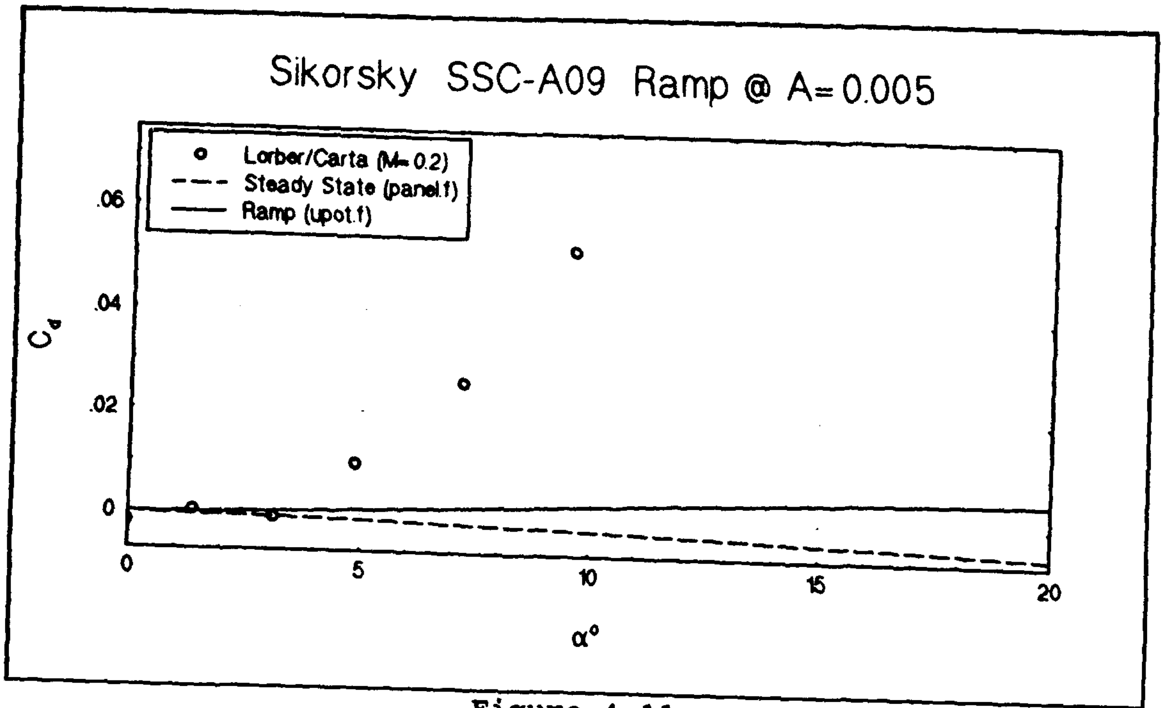


Figure 4.11

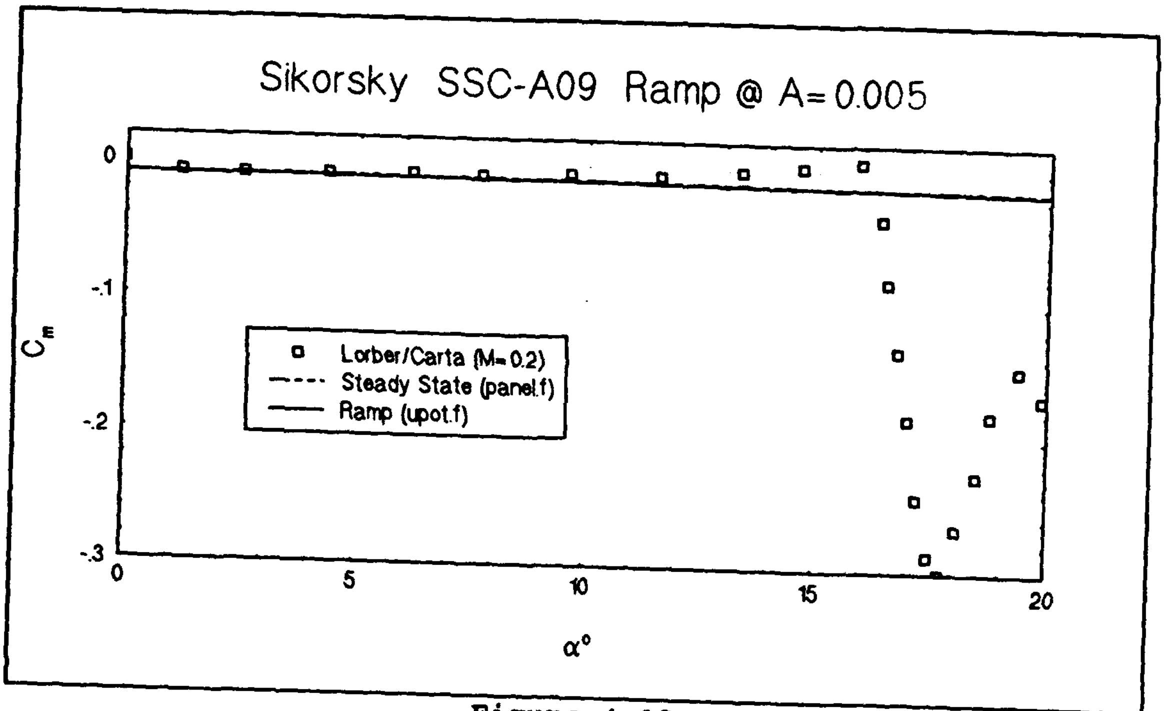


Figure 4.12

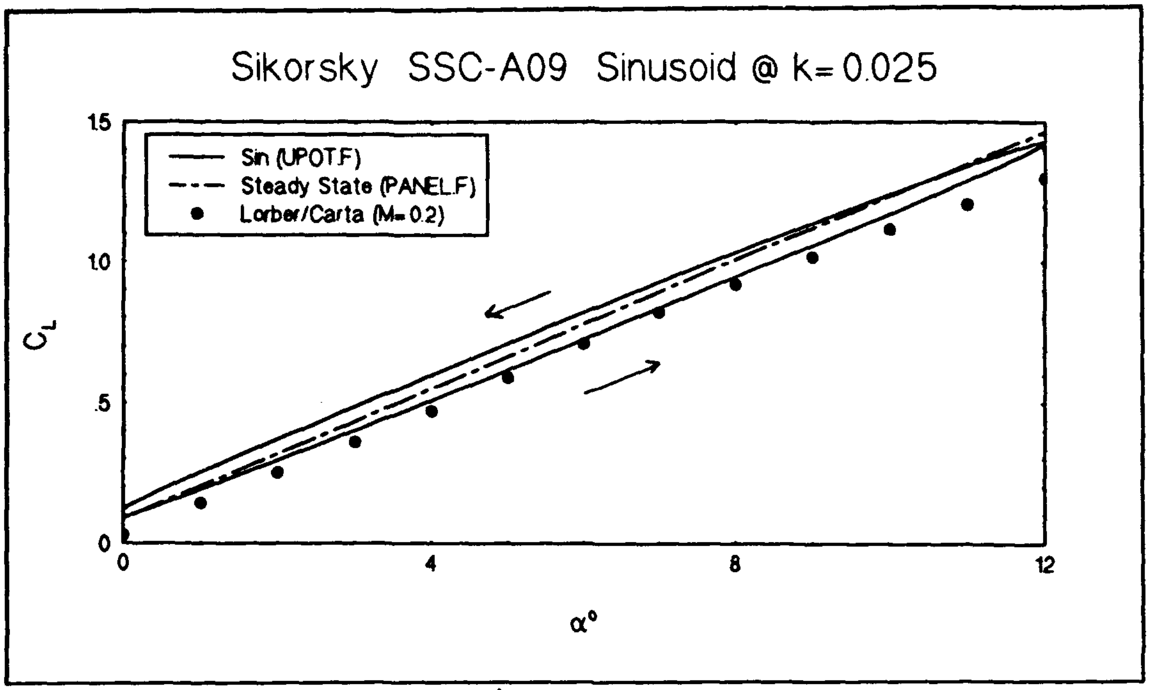


Figure 4.13

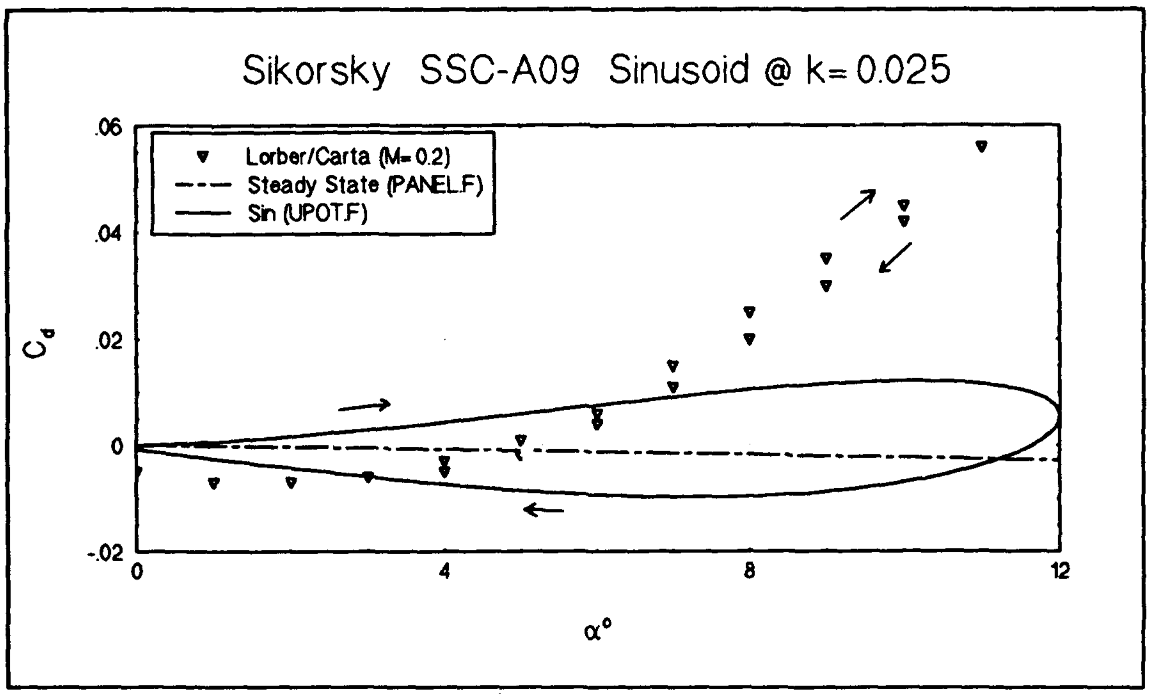


Figure 4.14

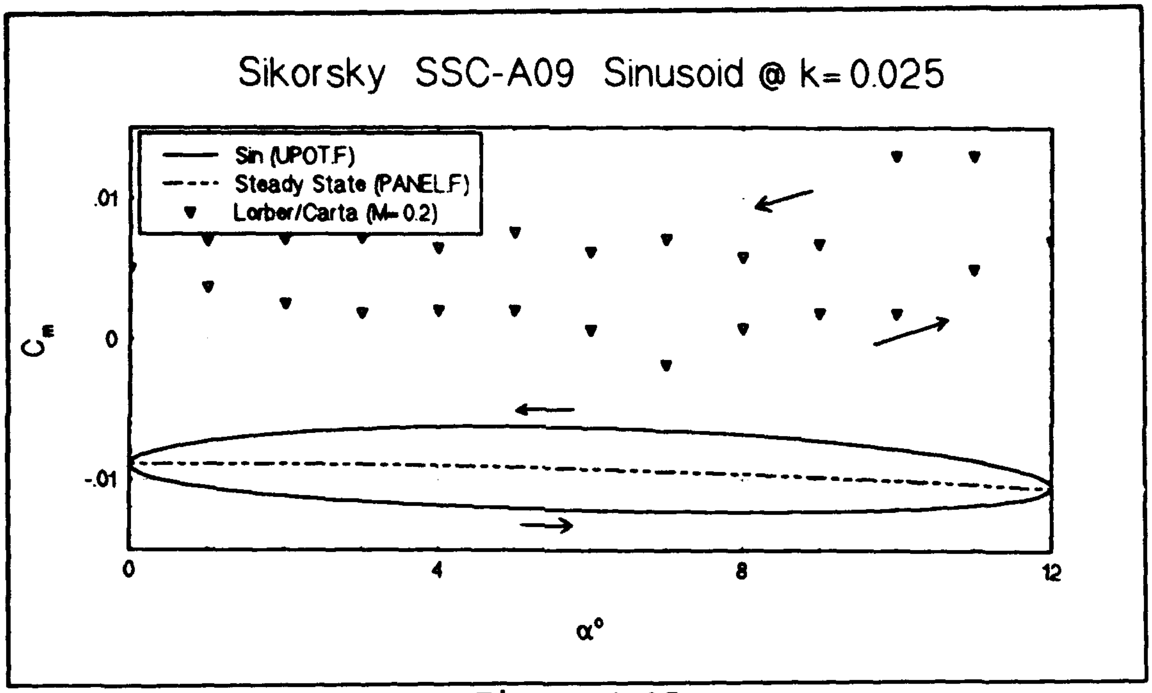


Figure 4.15

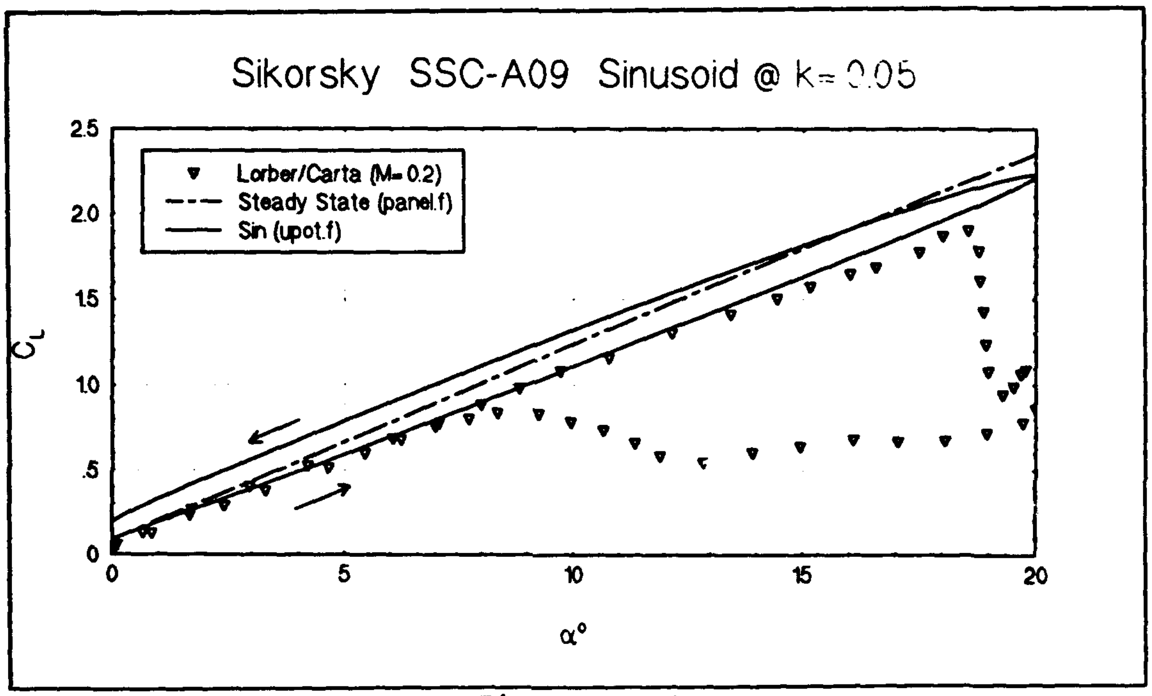


Figure 4.16

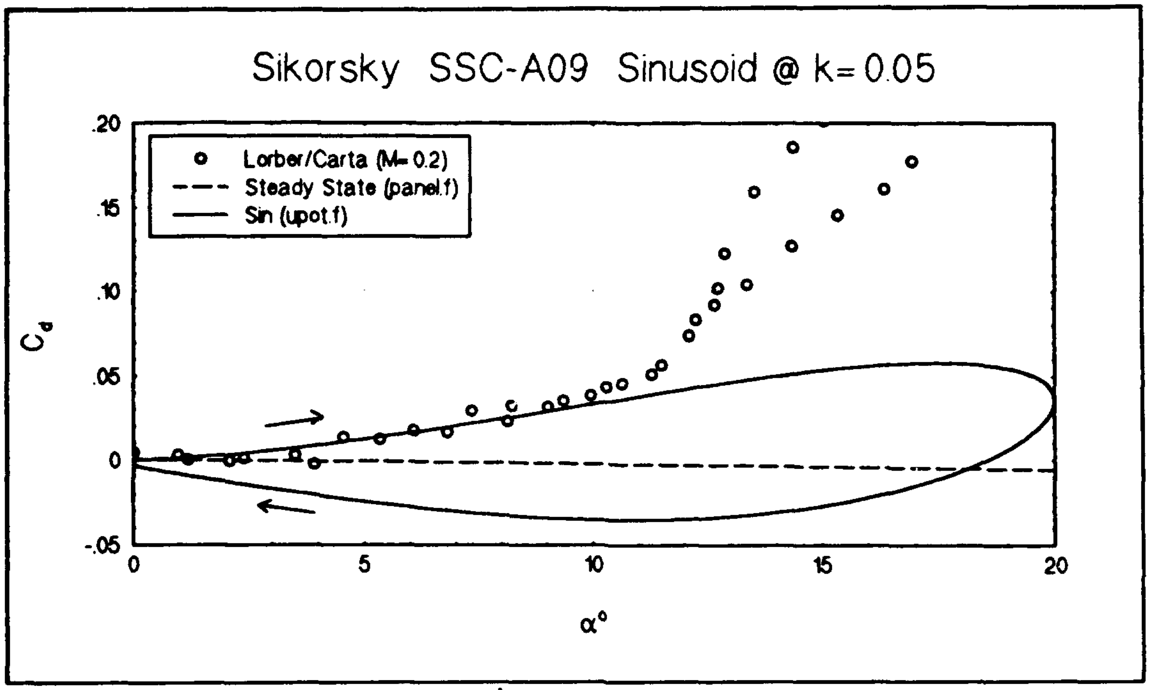


Figure 4.17

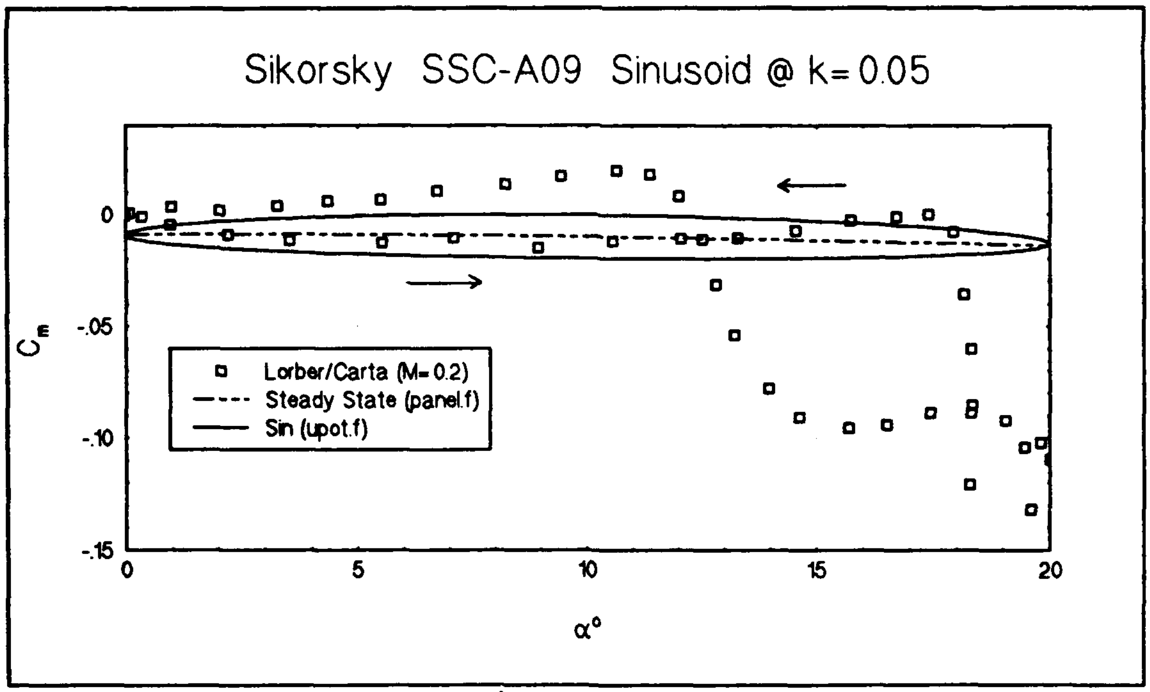


Figure 4.18

V. NAVIER-STOKES CODE

A. THEORY/BACKGROUND

To fully understand and visualize the viscous and compressibility effects on the dynamic stall phenomenon, a numerical solution of the unsteady Navier-Stokes equations is required. A thorough understanding of the flow physics at the leading edge region is most important due to the presence of significant compressibility effects and boundary layer transition. Compressibility effects appear at 0.2 to 0.3 Mach numbers on the NACA 0012; and shocks can form on the airfoil upper surface at 0.45 Mach number. Interaction of the shock and the local boundary layer then directly affects the flow separation process (VanDyken and Chandrasekhara [Ref. 17]).

Dynamic stall is the fluid aerodynamic response to an airfoil executing a time-dependent pitch. Rapid pitch up generates a vortex near the leading edge that increases flow circulation and, therefore, lift. At angels-of-attack beyond the static stall angle-of-attack, massive unsteady separation and large-scale vortical structures characterize the unsteady flowfield (Srinivasan, Ekaterinaris, and McCroskey [Ref. 14]). Vortex convection aft along the airfoil creates large force and moment changes. Successive weaker vortices may be generated with continued pitching or oscillatory motion. The

flow completely reattaches only after the angle of incidence is significantly reduced. Experimental results of unsteady flows over pitching airfoils by Lorber and Carta [Ref. 11] produced supersonic speeds and generated shocks near the leading edge at M=0.3 and higher free stream Mach numbers. The dynamic stall flowfield dependence parameters are: airfoil shape, Mach number, reduced frequency or reduced pitch rate, oscillation amplitude motion type (ramp or sinusoid), Reynolds number, and wind tunnel wall effects (Srinivasan, Ekaterinaris, and McCroskey [Ref. 14]). Wind tunnel wall effects were not included in this investigation.

B. NUMERICAL SCHEME

The numerical scheme and implementation used in this paper were taken from one developed by Tuncer, Ekaterinaris, and Platzer [Ref. 16] and Cricelli, Ekaterinaris, and Platzer [Ref. 6]. The strong, conservation law form of the two-dimensional, thin-layer Navier-Stokes equations is used. For a curvilinear coordinate system, (ξ,ζ), along streamwise and normal direction respectively, the governing equations take the following form:

$$\partial_t\hat{Q} + \partial_\xi\hat{F} + \partial_\zeta\hat{G} = R_e^{-1}\,\partial_\zeta\hat{S} \qquad (5.1)$$

Here **Q**, Equation 5.2, is the vector of the conservative variables. The inviscid flux vectors, **F** and **G**, are shown in

Equation 5.3. U and W are contravariant velocity components given by Equation 5.4. For the thin-layer approximation of the viscous flux term $\boldsymbol{S}$, in the ζ direction normal to the airfoil surface is shown in Equation 5.5.

$$\hat{Q} = \frac{1}{J}\begin{Bmatrix} \rho \\ \rho u \\ \rho w \\ e \end{Bmatrix} \tag{5.2}$$

$$\hat{F} = \frac{1}{J}\begin{Bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho w U + \xi_z p \\ (e + p)U - \xi_t p \end{Bmatrix}$$
$$\hat{G} = \frac{1}{J}\begin{Bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho w W + \zeta_z p \\ (e + p)W - \zeta_t p \end{Bmatrix} \tag{5.3}$$

$$\begin{aligned} U &= u\xi_x + w\xi_z + \xi_t \\ W &= u\zeta_x + w\zeta_z + \zeta_t \end{aligned} \tag{5.4}$$

$$
\hat{S} = \frac{1}{J}\left\{\begin{array}{c} 0 \\ \mu m_1 u_\zeta + \left(\frac{\mu}{3}\right) m_2 \zeta_x \\ \mu m_1 w_\zeta + \left(\frac{\mu}{3}\right) m_2 \zeta_z \\ \mu m_1 m_3 + \left(\frac{\mu}{3}\right) m_2 + (\zeta_x u + \zeta_z w) \end{array}\right\} \tag{5.5}
$$

$$
m_1 = \zeta_x^2 + \zeta_z^2
$$

$$
m_2 = \zeta_x u_\zeta + \zeta_z w_\zeta
$$

$$
m_3 = \frac{u^2+w^2}{2} + \kappa P_r^{-1}\left(\frac{\partial a^2}{\partial \zeta}\right)
$$

In Equations 5.1 through 5.5, all geometrical dimensions are normalized by the root-chord length; density is normalized by the free-stream density, ρ_∞; u and w are the Cartesian velocity components of the physical domain normalized by the free-stream speed of sound, a_∞; e is the total energy per unit volume normalized by $\rho_\infty a^2_\infty$; and P_r is the Prandtl number. The equation of state for an ideal gas relates pressure to density and total energy and is presented in Equation 5.6. The flow field is assumed to be fully turbulent (the laminar and transitional boundary layers are neglected) and the Baldwin-Lomax turbulence model, Section III.A.1, is used to evaluate eddy viscosity.

$$p = (\gamma-1)\left[e - \frac{\rho(u^2+w^2)}{2}\right] \tag{5.6}$$

1. Boundary Conditions

The computational domain includes the airfoil and the entire viscous flow field. The no-slip boundary condition is applied on the airfoil surface. The density and pressure values are obtained by extrapolation. If the flow is unsteady, the surface fluid velocity is set to the dictated airfoil velocity to satisfy the no-slip boundary condition (Tuncer et al, [Ref. 16]).

Flow variables are evaluated using the zero-order Riemann invariant extrapolation at the downstream outflow boundary. Only pressure, the incoming characteristic, is specified at a subsonic outflow boundary and three outgoing characteristics are extrapolated from the interior. A first order extrapolation is used for density and normal velocity. The zero-order outgoing Riemann invariant determines the axial flow velocity. The outflow boundary conditions are shown in Equation 5.7.

$$\begin{aligned}
\rho_1 &= \rho_2 \\
u_1 &= R_1^+ - \frac{2a_1}{\gamma-1}, \qquad a_1 = \sqrt{\frac{\gamma\, p_1}{\rho_1}} \\
w_1 &= w_2 \\
p_1 &= p_\infty
\end{aligned} \tag{5.7}$$

2. Numerical Implementation

The numerical integration is obtained using a high-order accurate upwind biased, factorized, iterative, implicit scheme developed by Tuncer et al [Ref. 16]. The inviscid fluxes are evaluated using Osher's third-order-accurate upwind scheme. Time accuracy of the implicit numerical solution is improved by Newton subiteration within each time step. Complete code details are located in Tuncer et al [Ref. 16] and Cricelli et al [Ref. 5].

a. *NPS Cray Y-MP*

All NS.F program runs were accomplished using the Naval Postgraduate School Cray Y-MP EL computer. It is a 133 MFLOP processor with 2GB of main memory and a 50 GB local disk running Unicos 7.

b. *PLOT3D*

The solutions generated by any CFD program consist of millions of numbers representing grid points and physical variable magnitudes. Visual techniques are relied upon to translate this vast numerical data base into comprehensible graphic representations. PLOT3D is a computer graphics program that allows interactive fluid dynamics examination. Physical phenomena are represented by color gradations and through individual particle traces. Program PLCON.F, Appendix C, is used to write the visualization output files that are displayed here.

3. Computational Grid

A 213 by 61 point body-fitted, C-type computational grid was used in all computations. The grid had 213 points around the airfoil (the trailing edge lower point at 31 and upper point at 183) and 61 points in the normal direction, all generated by a hyperbolic grid generator. The grid was clustered at the body surface in the normal direction, at the leading edge and at the trailing edge regions. The C-type grid provided a sufficiently high enough grid density at the airfoil surface boundary to resolve wall viscous and vortical flow field effects, as well as capturing the leading edge shock created during unsteady maneuvers and created at high angle-of-attack in steady flows at greater than 0.3 Mach. Cell orthogonality was emphasized throughout the grid and facilitated solution convergence.

4. Program NS.F User's Guide

A complete NS.F user's guide is located in Appendix C. Included are the NS.IN input name list; Indigo and Cray Y-MP Batch and graphical interface codes; and the NS.F source code.

C. SIKORSKY SSC-A09 RESULTS

1. Steady State Motion

a. Force and Pitching-Moment Coefficients

Force and pitching-moment coefficient steady state results as a function of angle-of-attack for the Navier-Stokes, Panel code, and Lorber and Carta data for 0.2 and 0.4

Mach number flows are displayed in Figures 5.1 through 5.5. The Navier-Stokes code more closely approximates the lift coefficient experimental data through 14° angle-of-attack than the Panel code with accuracy decreasing with increasing Mach number. Some improvement is observed in moment coefficient calculations over the Panel code through 13° angle-of-attack. The Navier-Stokes code did calculate the experimentally measured rapid change in pitching-moment coefficient beyond 15° angle-of-attack with accuracy improving with increasing Mach number. The Navier-Stokes code closely followed experimentally measured drag results through 6° angle-of-attack, and accuracy improved with increasing Mach number beyond this point. The qualitative change in drag with increasing Mach number was properly calculated by the Navier-Stokes code.

b. Skin Friction Coefficient

Skin friction coefficient as a function of airfoil position, angle-of-attack, and Mach number are displayed in Figures 5.6 and 5.7. A 'loads' subroutine error in the NS.F code required modification of the displayed coefficients by a constant factor of 200. A new code version incorporating the Chen-Tysen flow transition model in a new 'bltrans' subroutine appears to have corrected this magnitude error. A separation bubble is observed at 0.2 Mach number located at approximately ten percent chord at both 9° and 11° angle-of-attack. The

trailing edge separation region grows and moves forward with increasing angle-of-attack. Separation bubbles also appear at 0.4 Mach number at 9°, 11°, and 13° angle-of-attack. The trailing edge separation region behaves the same, but is quantitatively smaller in size with increasing Mach number.

c. Pressure Coefficient

Pressure coefficient as a function of airfoil position, angle-of-attack, and Mach number is displayed in Figures 5.8 through 5.19. The Navier-Stokes code closely approximates the experimental data through 13° angle-of-attack with a small discrepancy at the trailing edge at low Mach numbers. Navier-Stokes correlation appears to be better with decreasing Mach number. The Baldwin-Lomax turbulence model was not able to accurately predict the shock induced boundary layer separation and reduced suction peak as can be observed in Figures 5.15, and 5.17 through 5.19.

d. Plot3D Visualization

Steady State density, pressure, Mach number, and vorticity flowfield variations as a function of Mach number and angle-of-attack are shown in Figures 5.20 through 5.31. The solutions presented for 0° angle-of-attack are the ones later used to initiate unsteady motion. At M=0.2, the flow remains attached until 9° α, at which point the trailing edge lightly separates and the separation region begins to move

forward with increasing angle-of-attack. At 15° α, the separation region extends forward to approximately 0.1 x/c.

At M=0.4, the flow again remains attached until 9° α, at which point the trailing edge separates a little more and the separation region begins to move forward with increasing angle-of-attack. Vorticity appears to increase with increasing Mach number. Shock induced boundary layer separation appears first at 11° α and increases in magnitude through 15° α where massive flowfield separation is observed, Figure 5.31.

2. Unsteady Motion

The unsteady calculations were started from the steady state solution at the lowest angle-of-attack. For sinusoidal motion, the instantaneous angle-of-attack is shown in Equation 5.8. Definitions for Reduced Pitch Rate, A, and Reduced Frequency, k, are given in Equations 4.3 and 4.4, respectfully.

$$\alpha(t) = \frac{(\alpha_{max} - \alpha_{min})}{2} \times [1 - \cos(\omega t)] \qquad (5.8)$$

a. *Ramp Motion, M=0.2, α=0°→ 30°, and A=0.005*

Density residual history as a function of angle-of-attack is displayed in Figure 5.32 and indicates stable

results through 15° angle-of-attack. Force and pitching-moment coefficient results, unsteady panel, and Lorber and Carta experimental results are displayed in Figures 5.33 through 5.35. The Navier-Stokes code more closely calculates lift and pitching-moment coefficients through 17° angle-of-attack than the unsteady panel code. The Navier-Stokes code closely followed experimentally measured drag through 10° angle-of-attack, but was late in calculating the experimentally measured sudden rise in drag at 16° angle-of-attack.

(1) Flowfield Visualization. Flowfield density, pressure, Mach number, and vorticity are displayed in Figures 5.36 through 5.42. A smaller leading edge stagnation region and a little larger trailing edge separation region is evident when compared to steady state results. Ramp maximum leading edge Mach number is also slightly smaller at any given angle-of-attack. Massive flowfield separation is observed at 20° angle-of-attack, Figure 5.42.

b. Ramp Motion, M=0.3 & 0.4, α=0°→ 20°, and A=0.005

Force and pitching-moment coefficient results as a function of Mach number and angle-of-attack and Lorber and Carta experimental results are displayed in Figures 5.43 through 5.48. The Navier-Stokes code somewhat overestimates force and pitching-moment coefficients for all Mach numbers throughout the attached flow region, but did capture the

experimentally measured changes at stall with accuracy improving with increasing Mach number. Drag coefficient followed experiment closely through the linear region with accuracy improving with increasing Mach number.

(1) Flowfield Visualization. Flowfield density, pressure, Mach number, and vorticity for the M=0.4 ramp are displayed in Figures 5.49 through 5.55. Mach number over the leading edge is higher than steady state values. Shock induced boundary layer separation is observed starting at 11° angle-of-attack and is more massive and moves more forward than observed in steady state results. At 15° angle-of-attack, the upper surface is mostly separated and convecting vortices are observed departing the trailing edge.

The leading edge flowfield region of the M=0.4 ramp motion is magnified in Figures 5.56 through 5.63. These figures graphically display the complex leading edge flow physics of shock induced boundary layer separation. A secondary shock on the leading edge can also be identified in Figures 5.56, 5.57, and 5.60. The time dependent vortex shedding process can be observed in Figures 5.62 and 5.63.

c. Sinusoid, M=0.2, and k=0.025 [6 - 6cos(ωt)]

Density residual history as a function of angle-of-attack is displayed in Figure 5.64 and indicates stable results throughout the angle-of-attack range. Force and pitching-moment coefficient results are displayed with

unsteady panel and Lorber and Carta results in Figures 5.65 through 5.67. Navier-Stokes lift results more closely follow the experimental results; but the character of the results, decreased lift on the up stroke and augmented lift on the down stroke, are opposite to the expected results. Drag computations improved over the unsteady panel code. The general trend is to follow experiment, but there is an error associated with angle-of-attack. Pitching-moment follows the experiment well, but seems to have a constant error. The experimental pitching-moment results are all positive which would not be expected with this cambered airfoil.

(1) Flowfield Visualization. Flowfield density, pressure, Mach number, and vorticity are displayed in Figures 5.68 through 5.74 for both the up and down stroke. Trailing edge separation is first observed at 9° angle-of-attack. The flow generally remains well behaved throughout the cycle and no shocks are indicated. Higher local Mach numbers are observed on the down stroke when compared to the up stroke for the same angle-of-attack.

d. Sinusoid, M=0.2, and k=0.05 [10 - 10cos(ωt)]

Density residual history as a function of angle-of-attack is displayed in Figure 5.75 and indicates stable results below 13.5° angle-of-attack. Force and pitching-moment coefficient results are displayed in Figures 5.76 through 5.78. The Navier-Stokes code followed experimentally

measured lift well through 19° angle-of-attack. A large difference in lift was calculated during the down cycle between 20° and 9° angle-of-attack. The Navier-Stokes code followed experimentally measured drag well through 12° angle-of-attack. There appears to be good moment coefficient agreement through 12° angle-of-attack.

(1) Flowfield Visualization. Flowfield density, pressure, Mach number, and vorticity are displayed in Figures 5.79 through 5.91 for both the up and down stroke. For angles-of-attack below 13°, local leading edge Mach number is higher and the trailing edge separation region is larger on the down stroke when compared to the up stroke at the same angle-of-attack. Beginning at 15° angle-of-attack, the local leading edge Mach number is higher on the up stroke. Massive trailing edge separation is observed on the down stroke at 17° angle-of-attack and disappears at 15° angle-of-attack. The flowfield is massively separated with complex vortical structures at the maximum angle-of-attack, 20°. The leading edge flowfield region is magnified in Figures 5.92 through 5.94. These figures graphically display the complex leading edge flow physics of shock induced boundary layer separation.

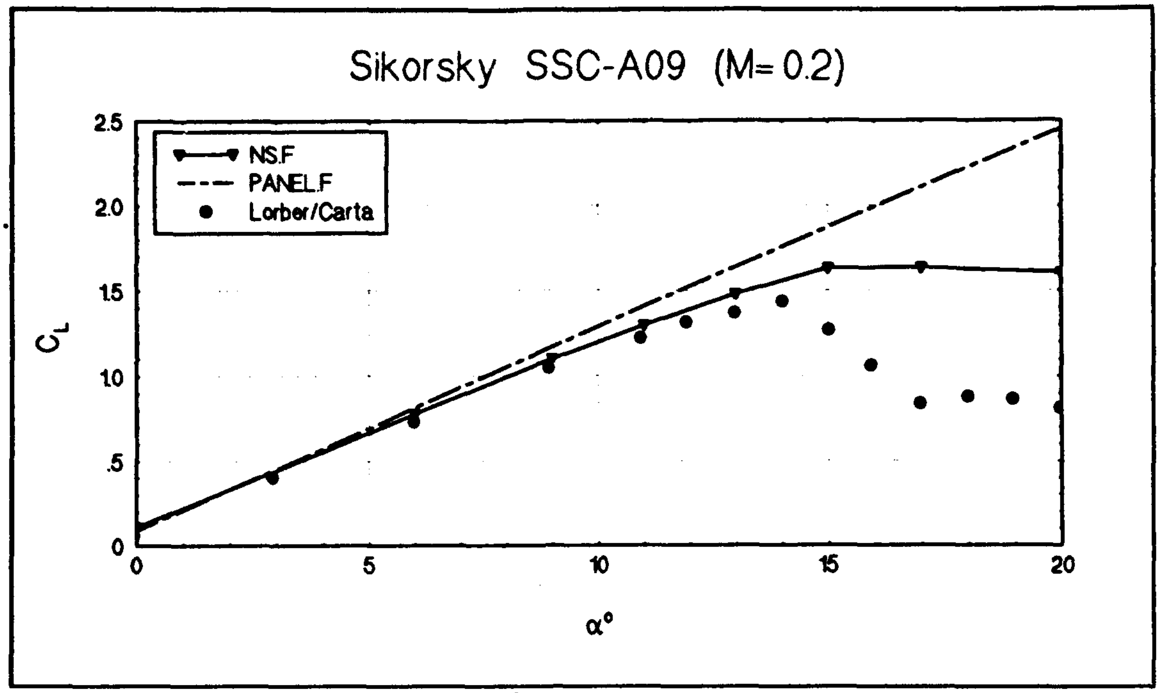


Figure 5.1
Steady State $C_{L\alpha}$ (M=0.2)

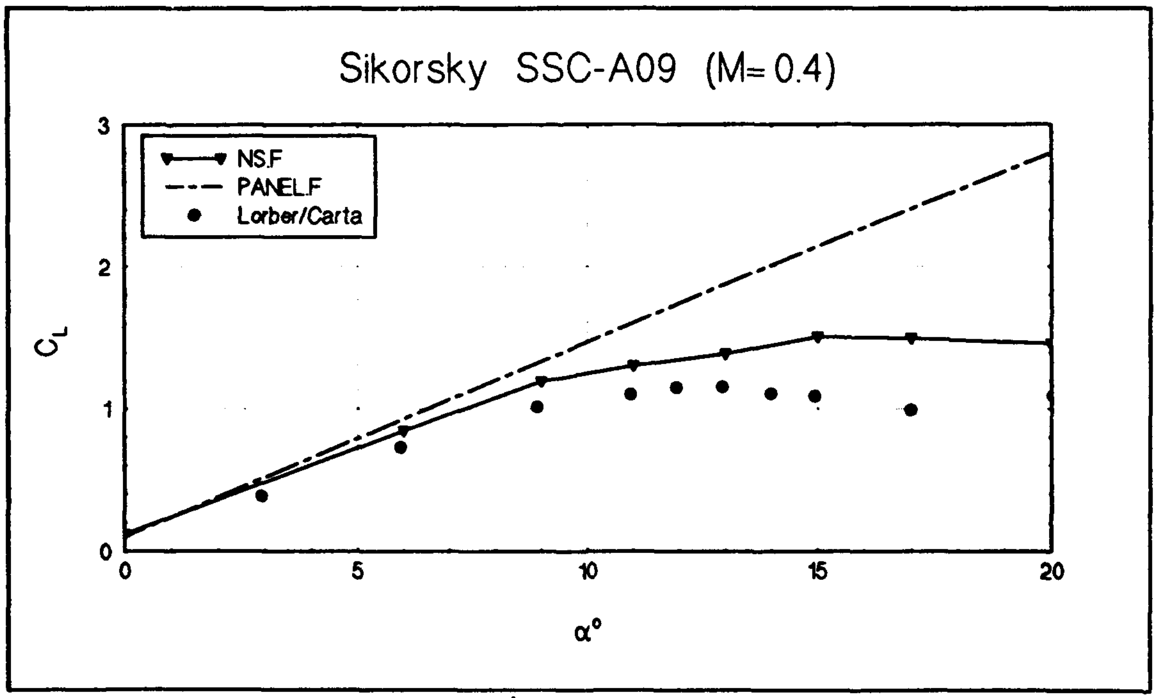


Figure 5.2
Steady State $C_{L\alpha}$ (M=0.4)

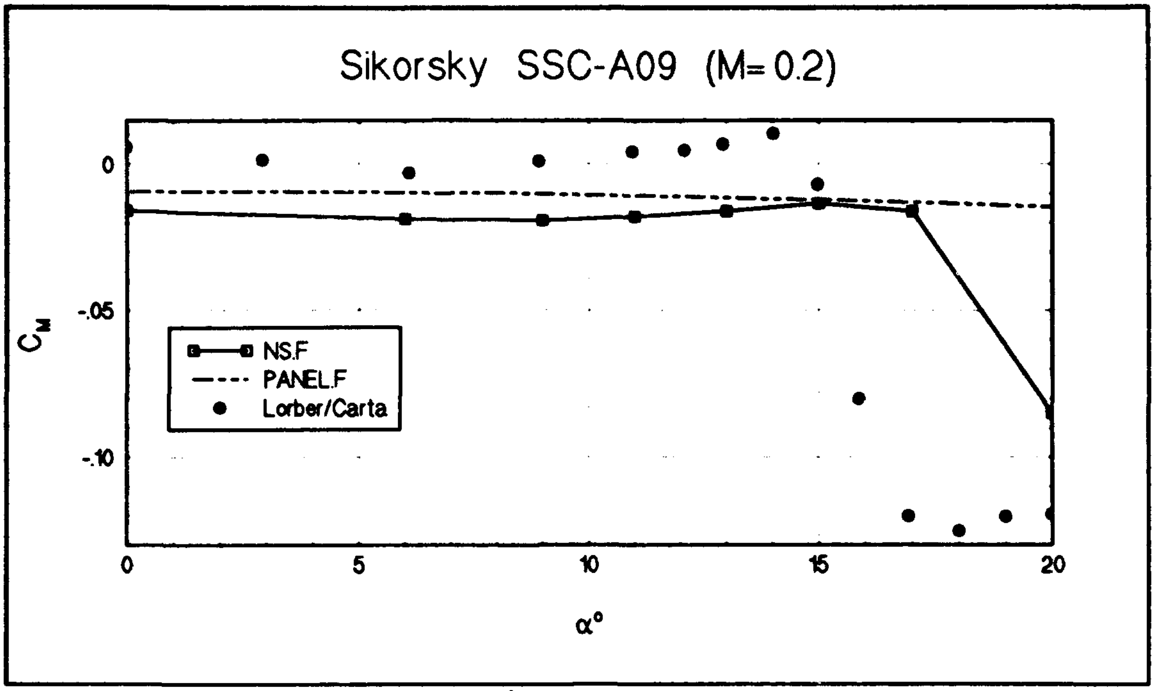


Figure 5.3
Steady State $C_{M\alpha}$ (M=0.2)

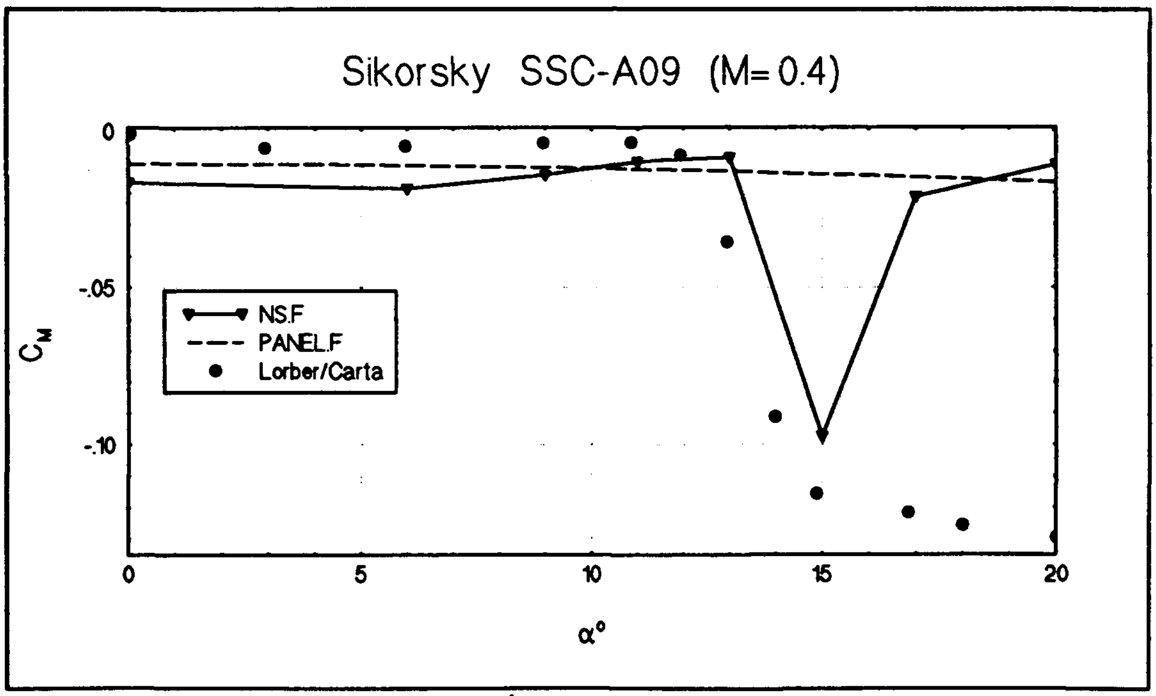


Figure 5.4
Steady State $C_{M\alpha}$ (M=0.4)

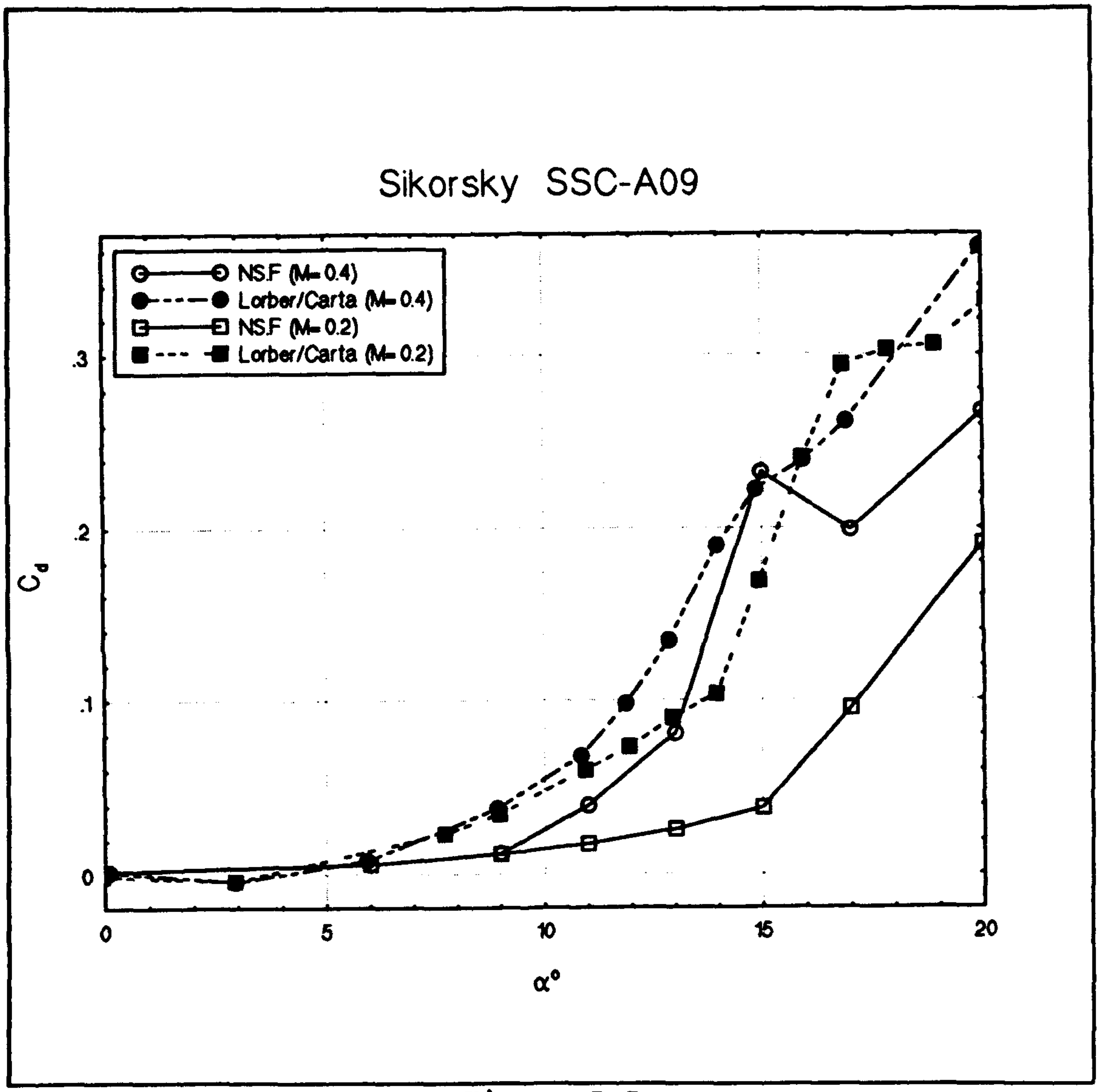


Figure 5.5
Steady State C_{d_s}

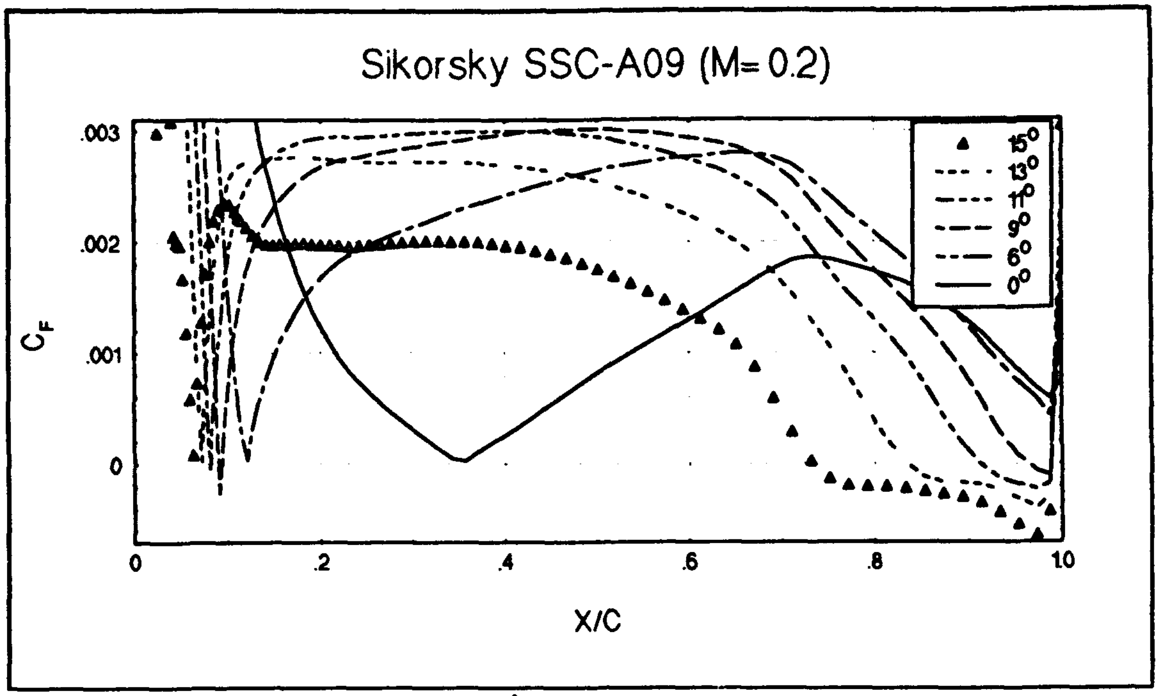


Figure 5.6
Steady State C_F (M=0.2)

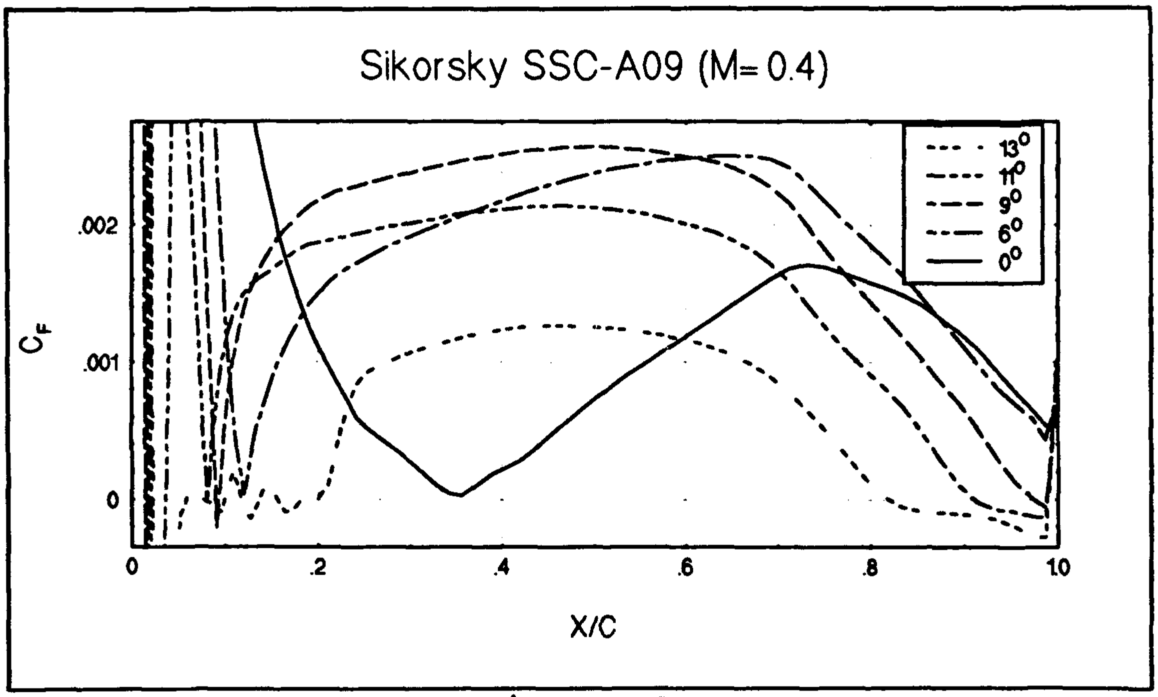


Figure 5.7
Steady State C_F (M=0.4)

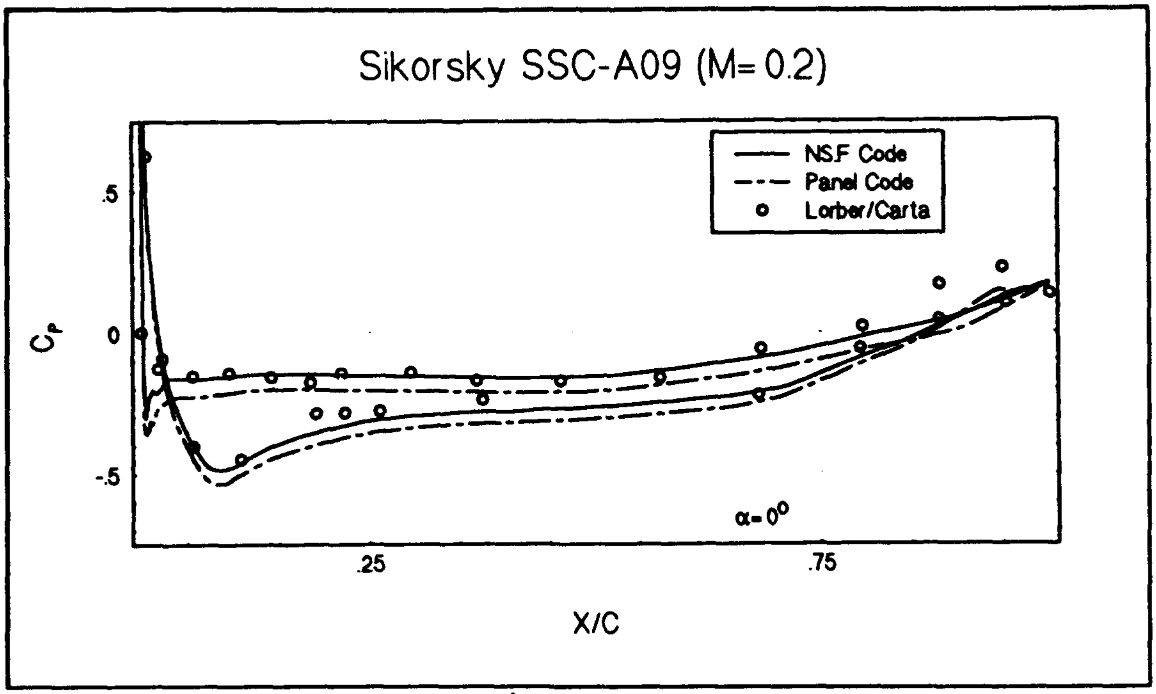


Figure 5.8
Steady State C_p (M=0.2)

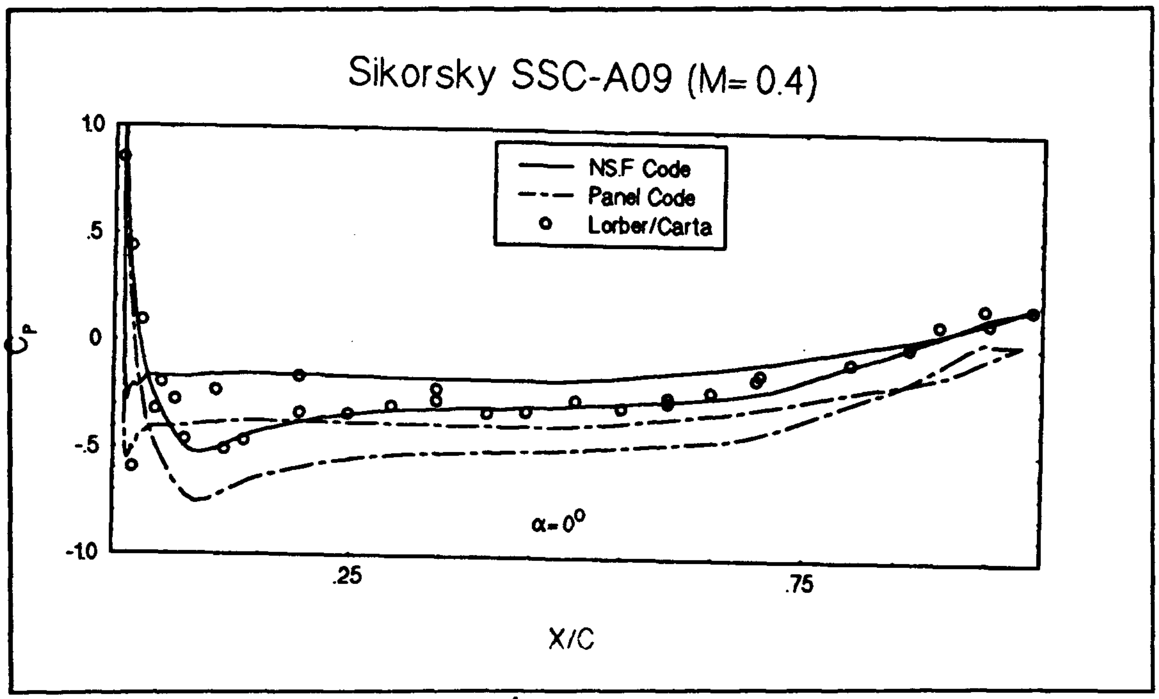


Figure 5.9
Steady State C_p (M=0.4)

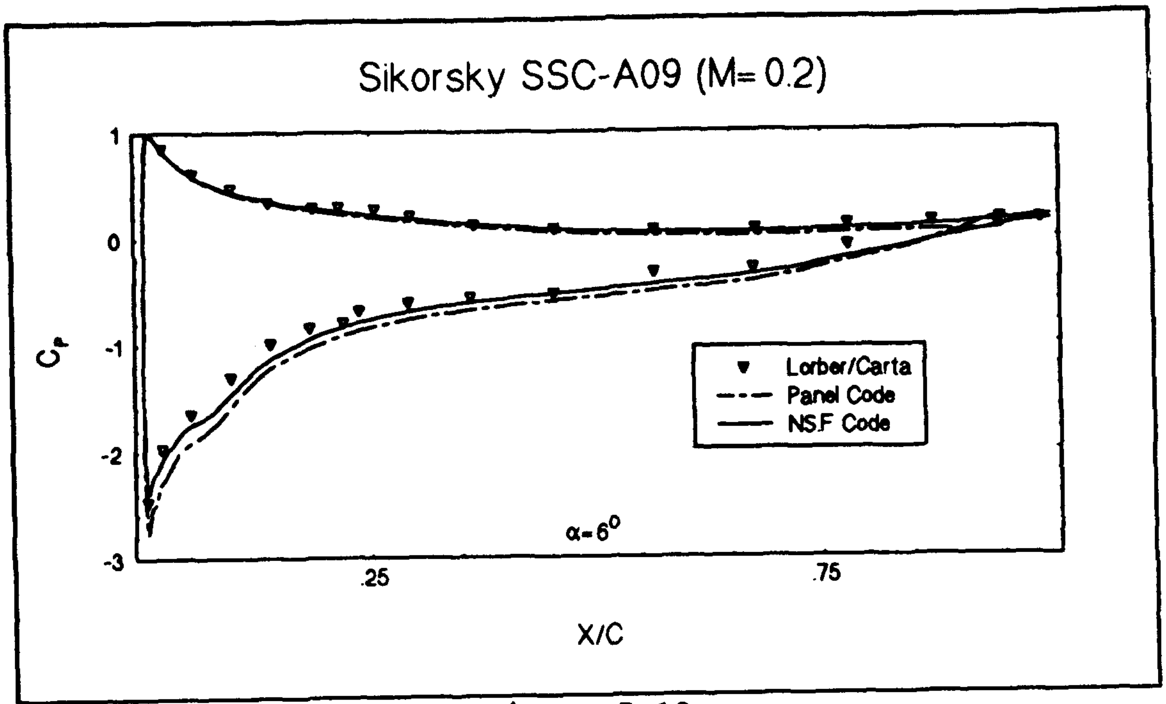


Figure 5.10
Steady State C_p (M=0.2)

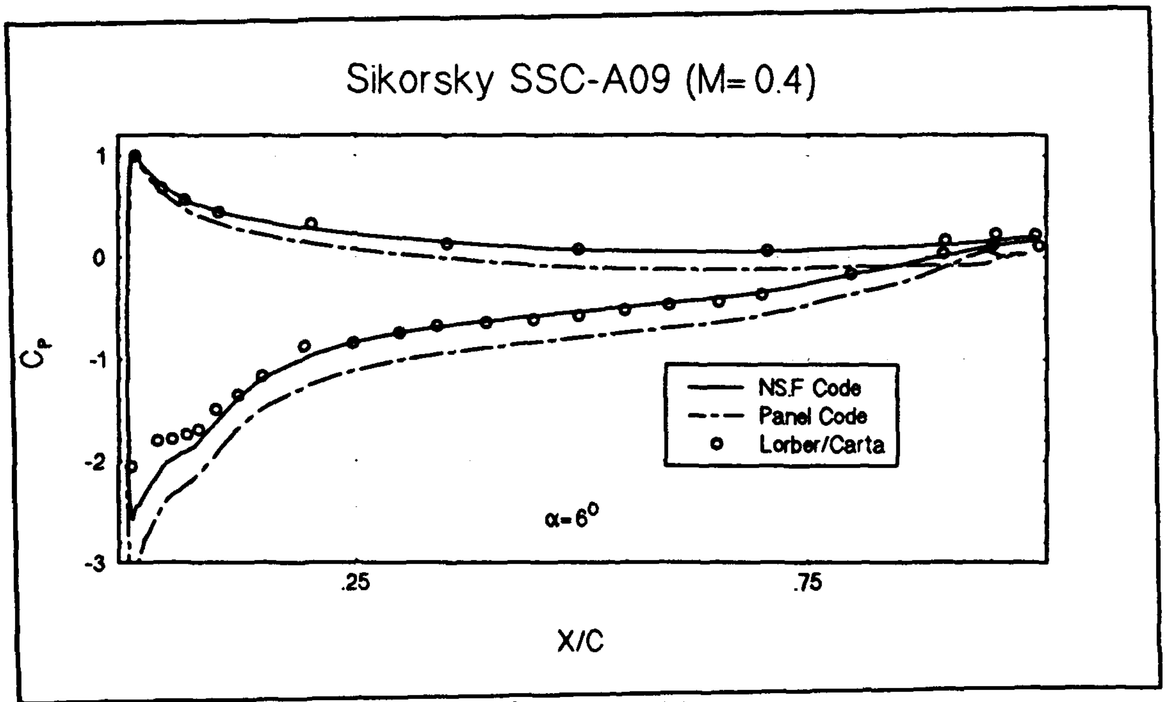


Figure 5.11
Steady State C_p (M=0.4)

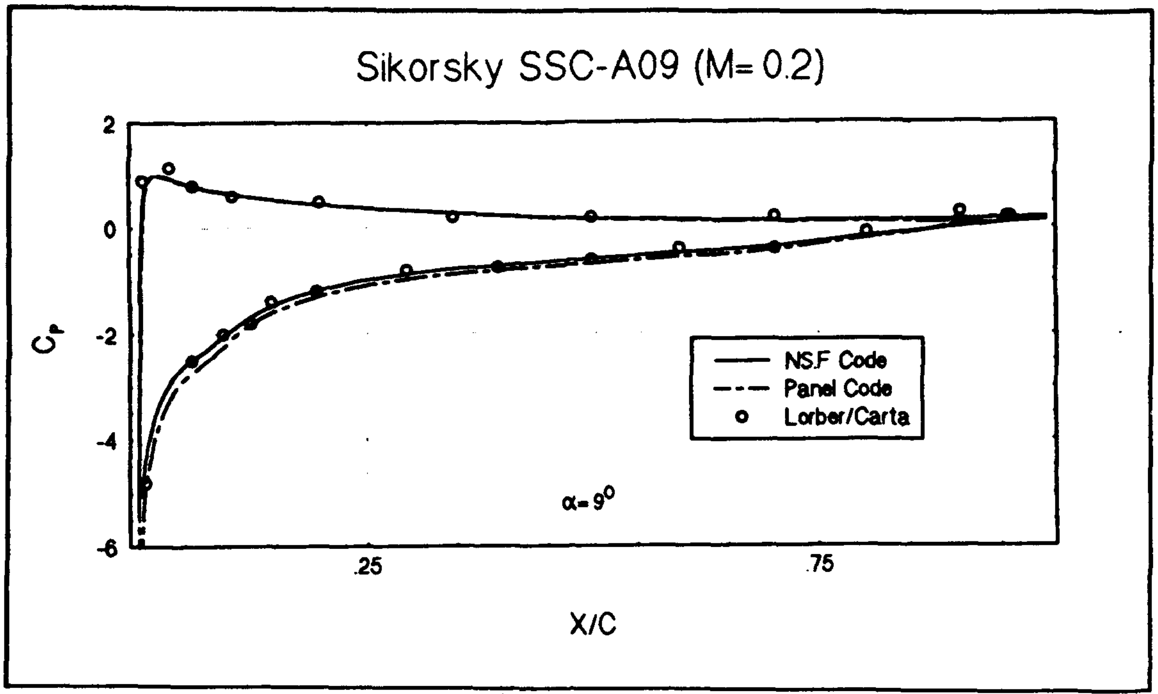


Figure 5.12
Steady State C_p (M=0.2)

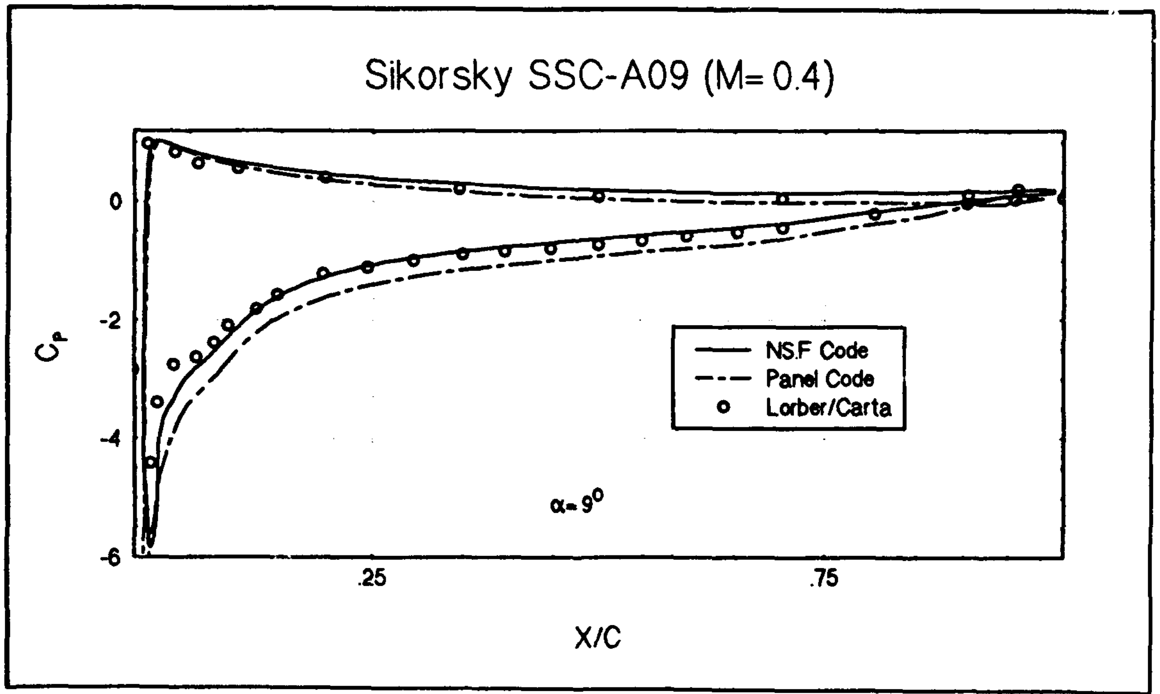


Figure 5.13
Steady State C_p (M=0.4)

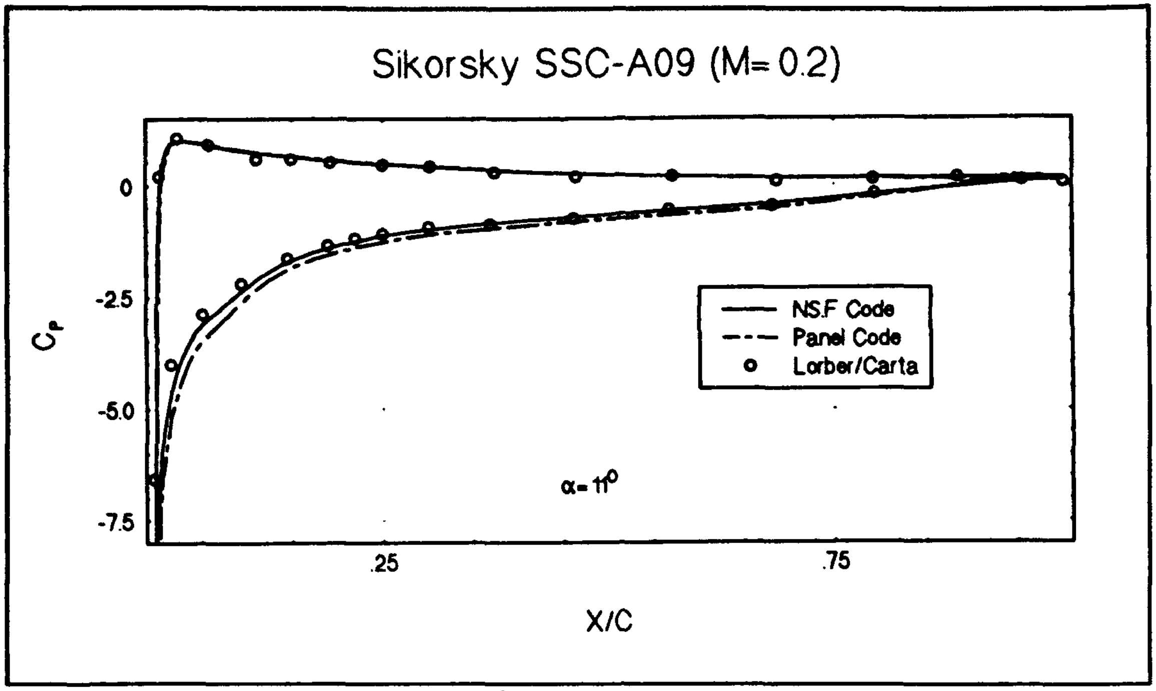


Figure 5.14
Steady State C_p (M=0.2)

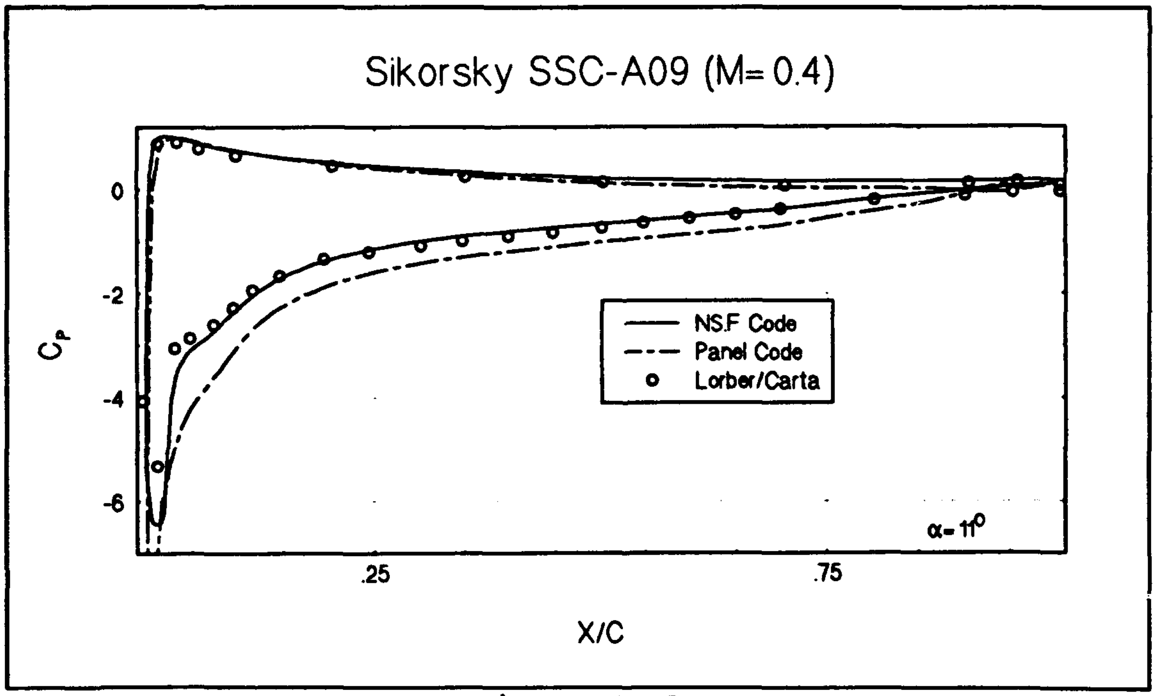


Figure 5.15
Steady State C_p (M=0.4)

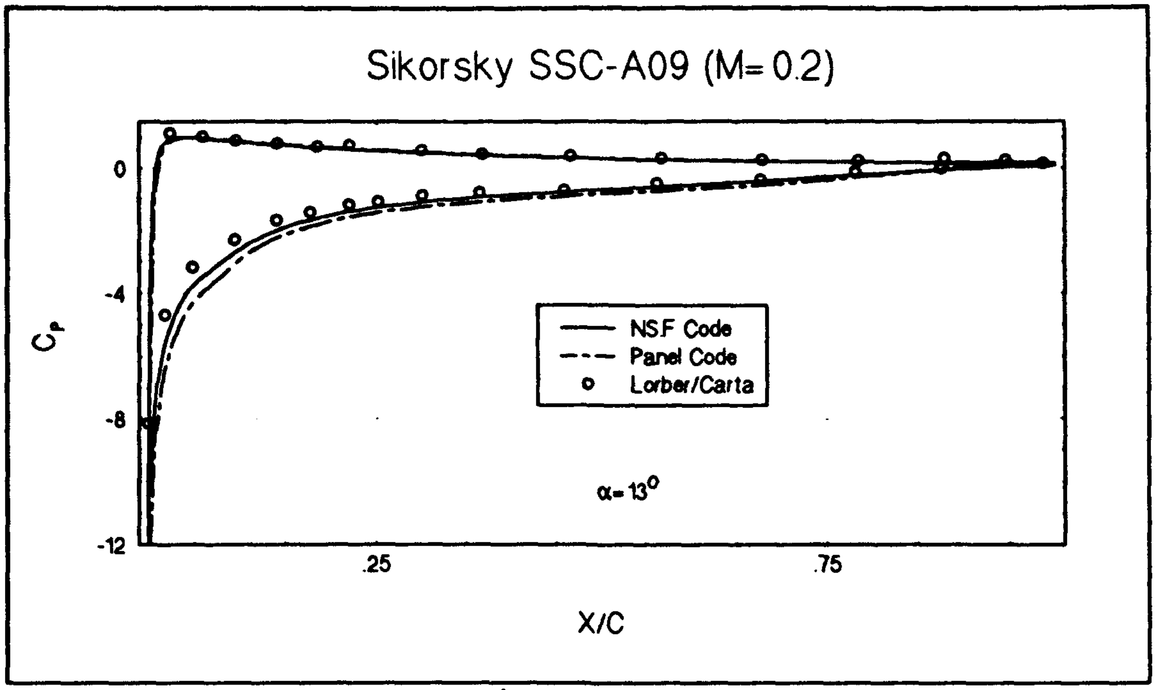


Figure 5.16
Steady State C_p (M=0.2)

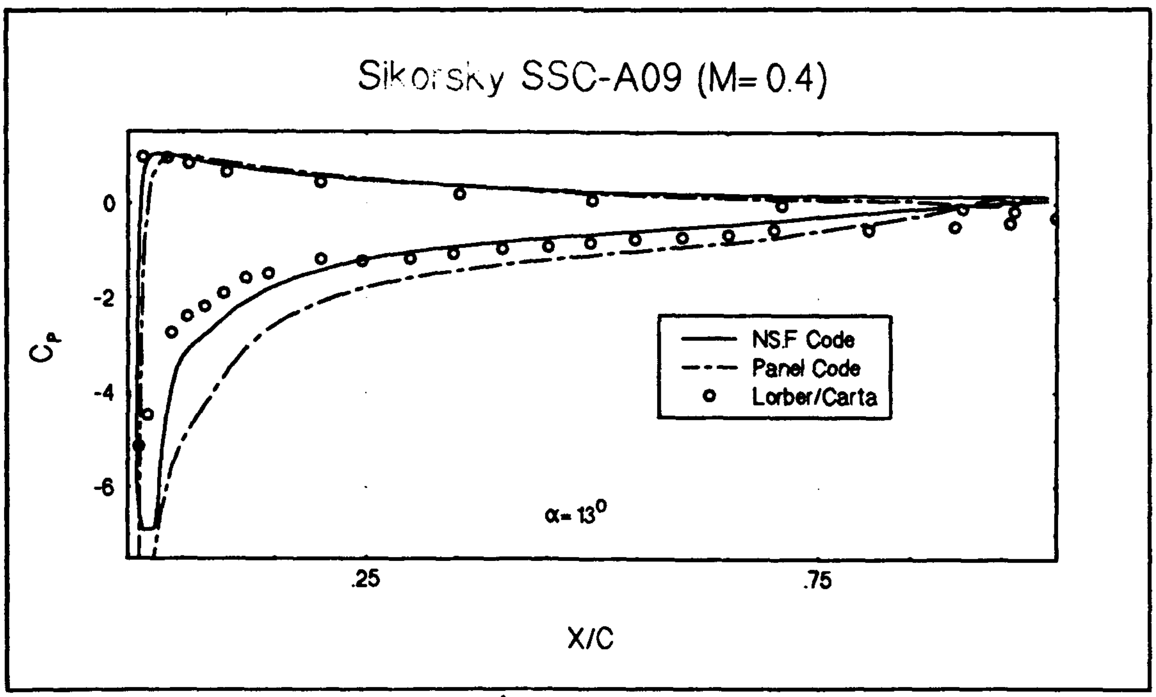


Figure 5.17
Steady State C_p (M=0.4)

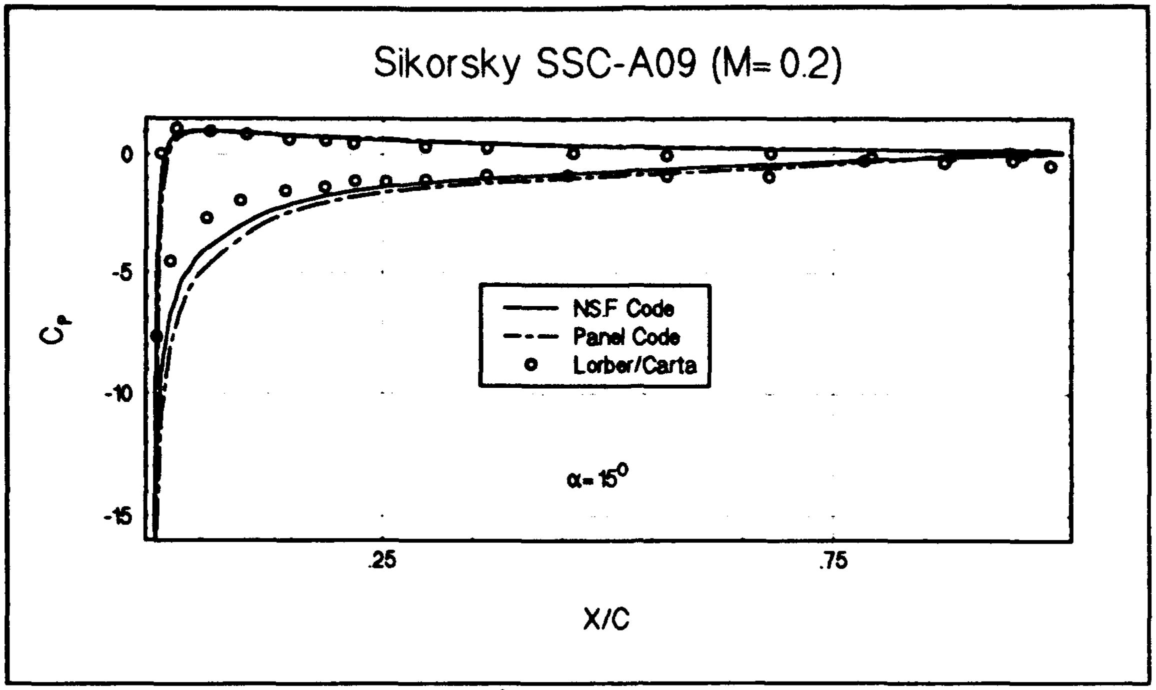


Figure 5.18
Steady State C_p (M=0.2)

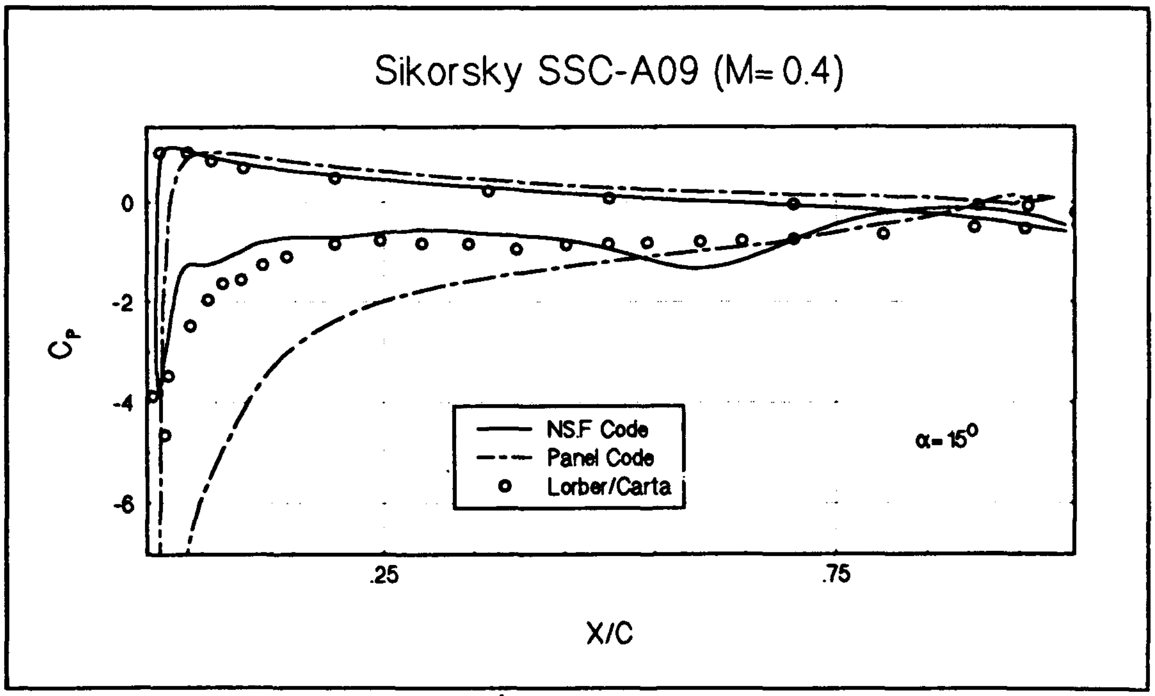


Figure 5.19
Steady State C_p (M=0.4)

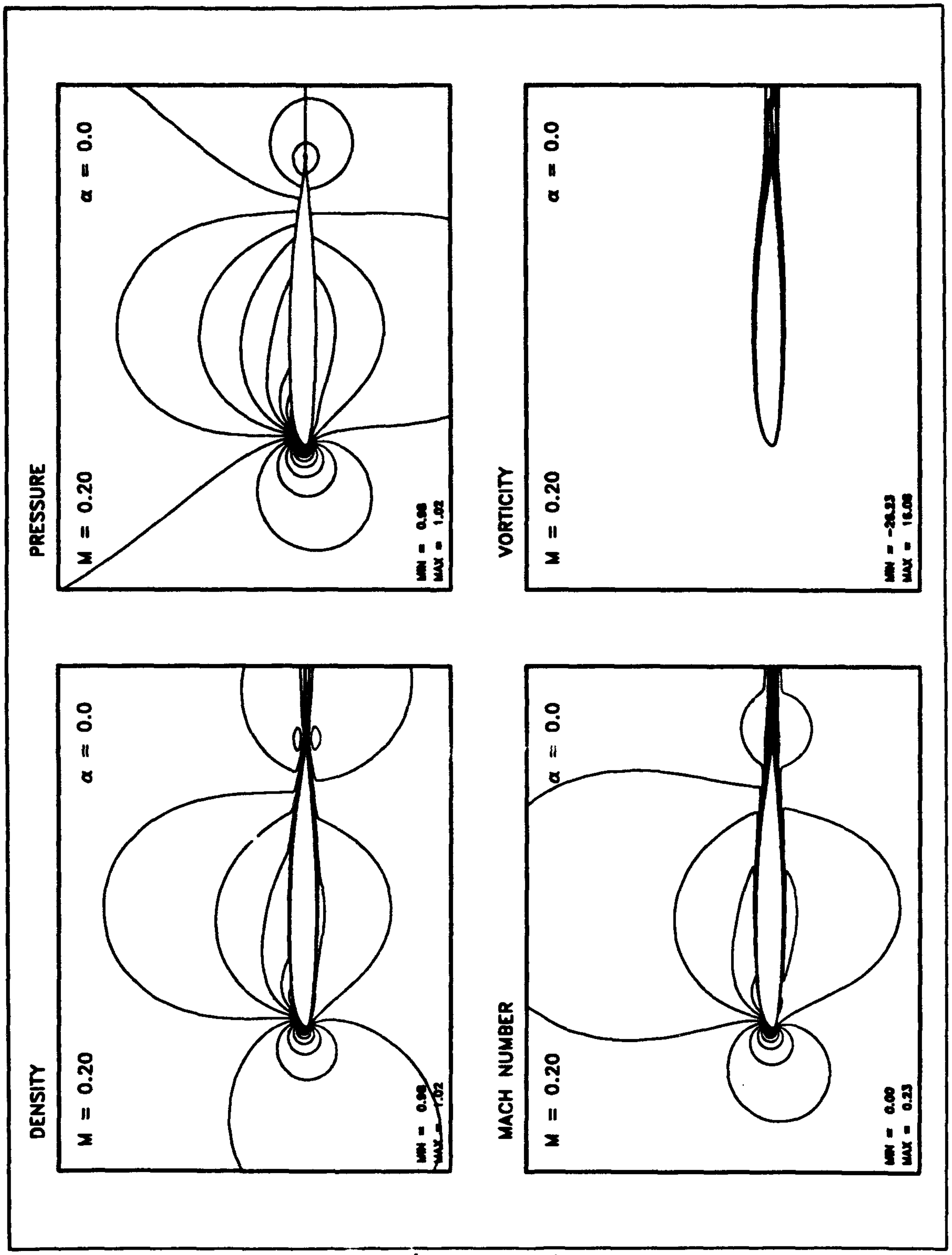


Figure 5.20
Sikorsky SSC-A09
Steady State (M=0.2)

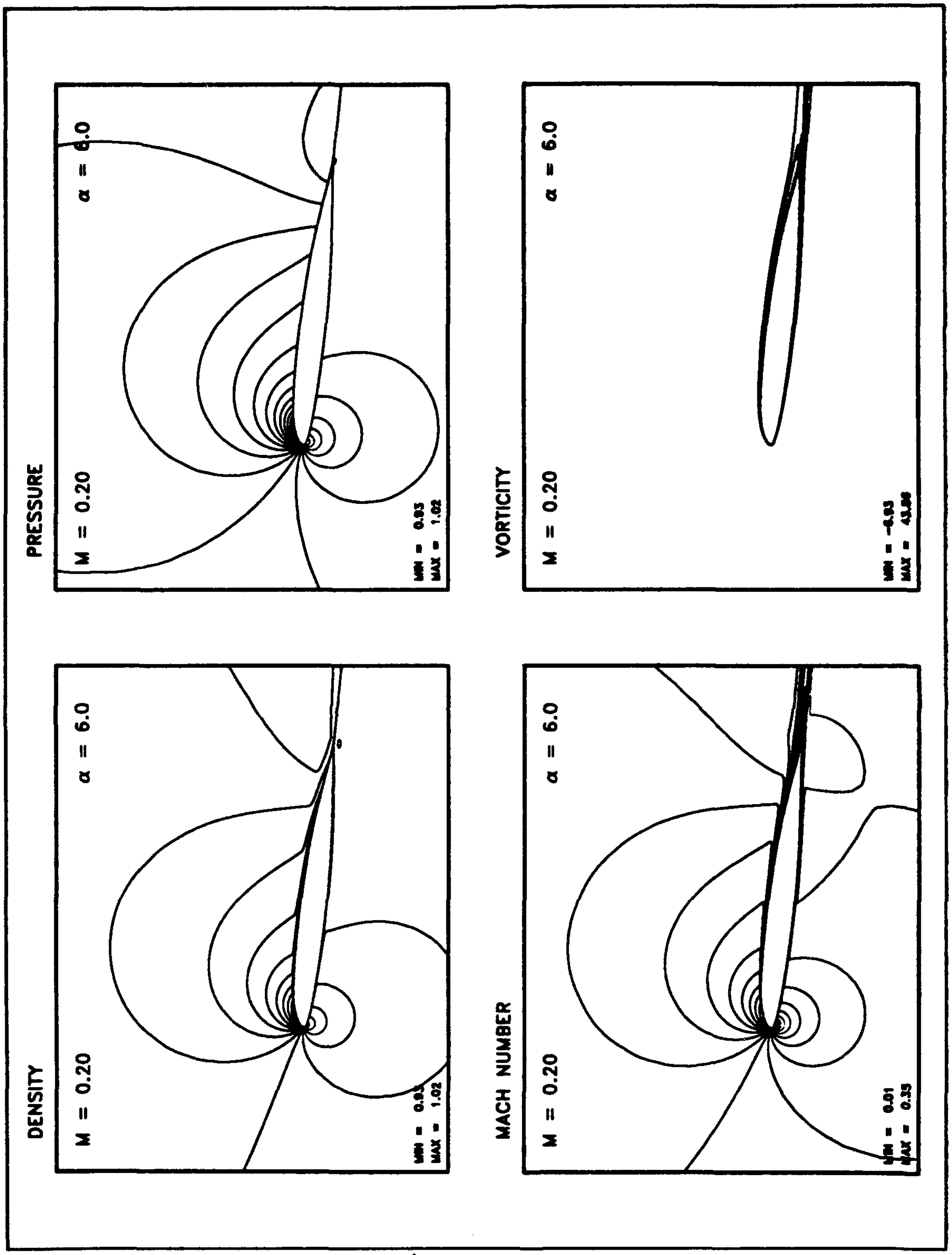


Figure 5.21
Sikorsky SSC-A09
Steady State (M=0.2)

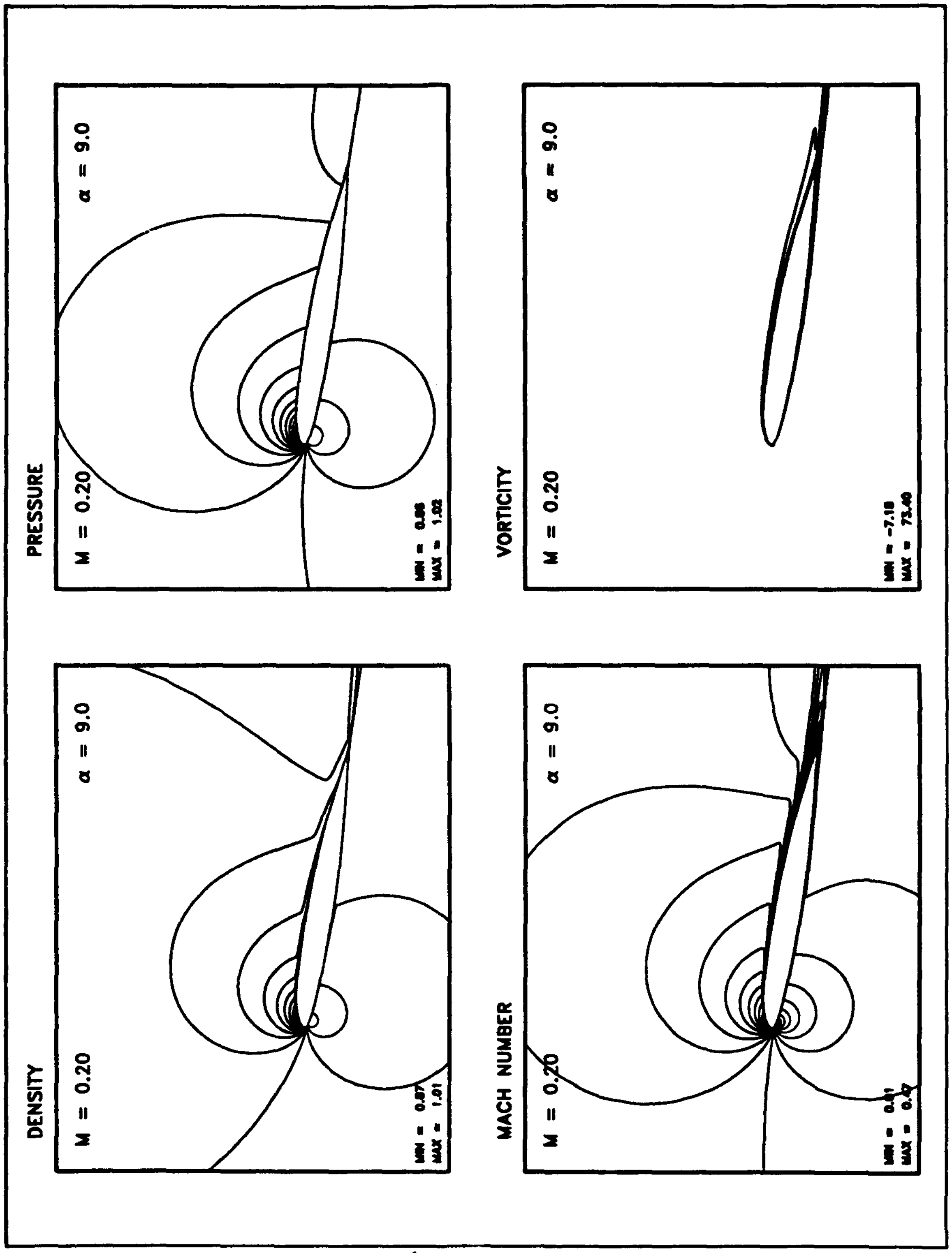


Figure 5.22
Sikorsky SSC-A09
Steady State (M=0.2)

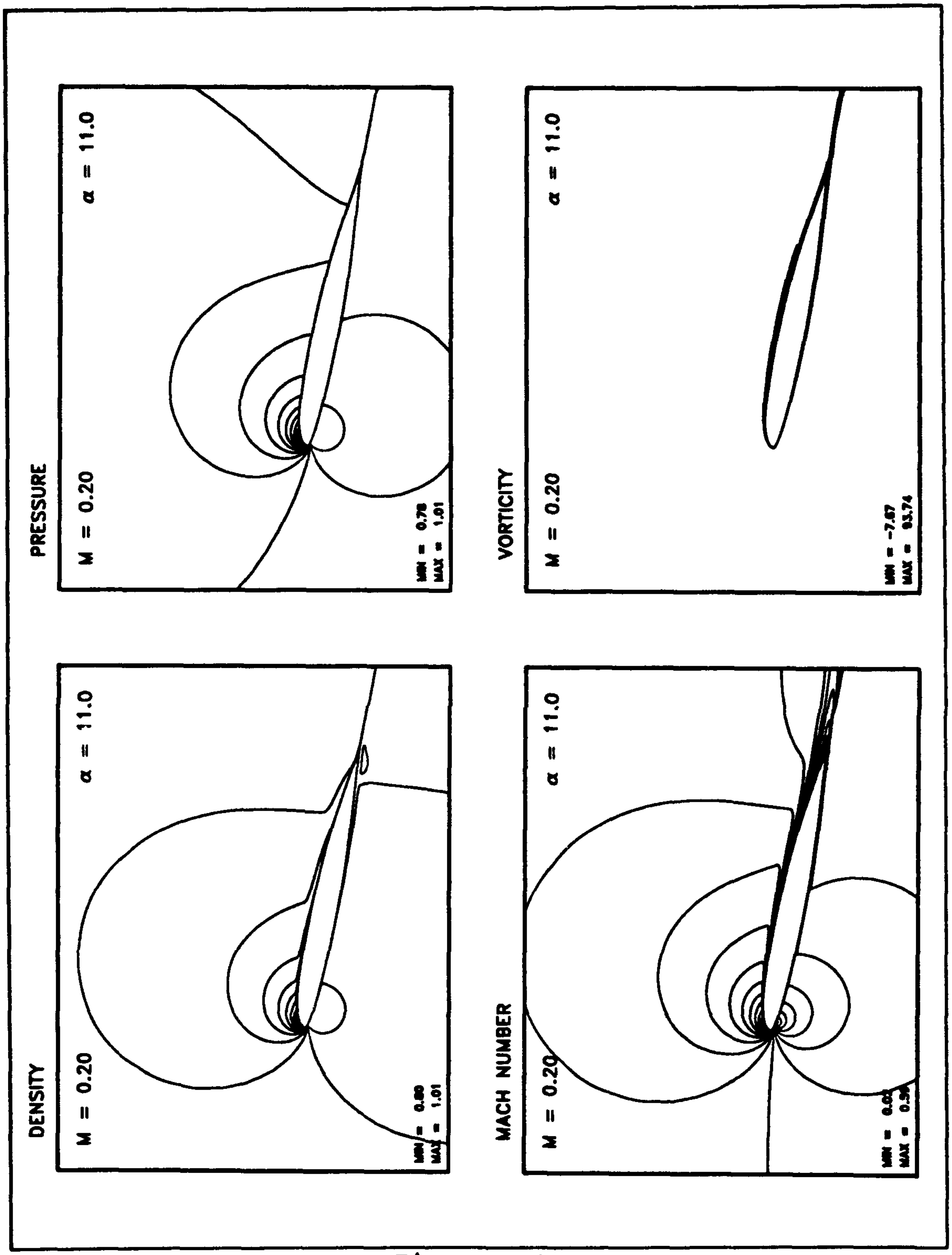


Figure 5.23
Sikorsky SSC-A09
Steady State (M=0.2)

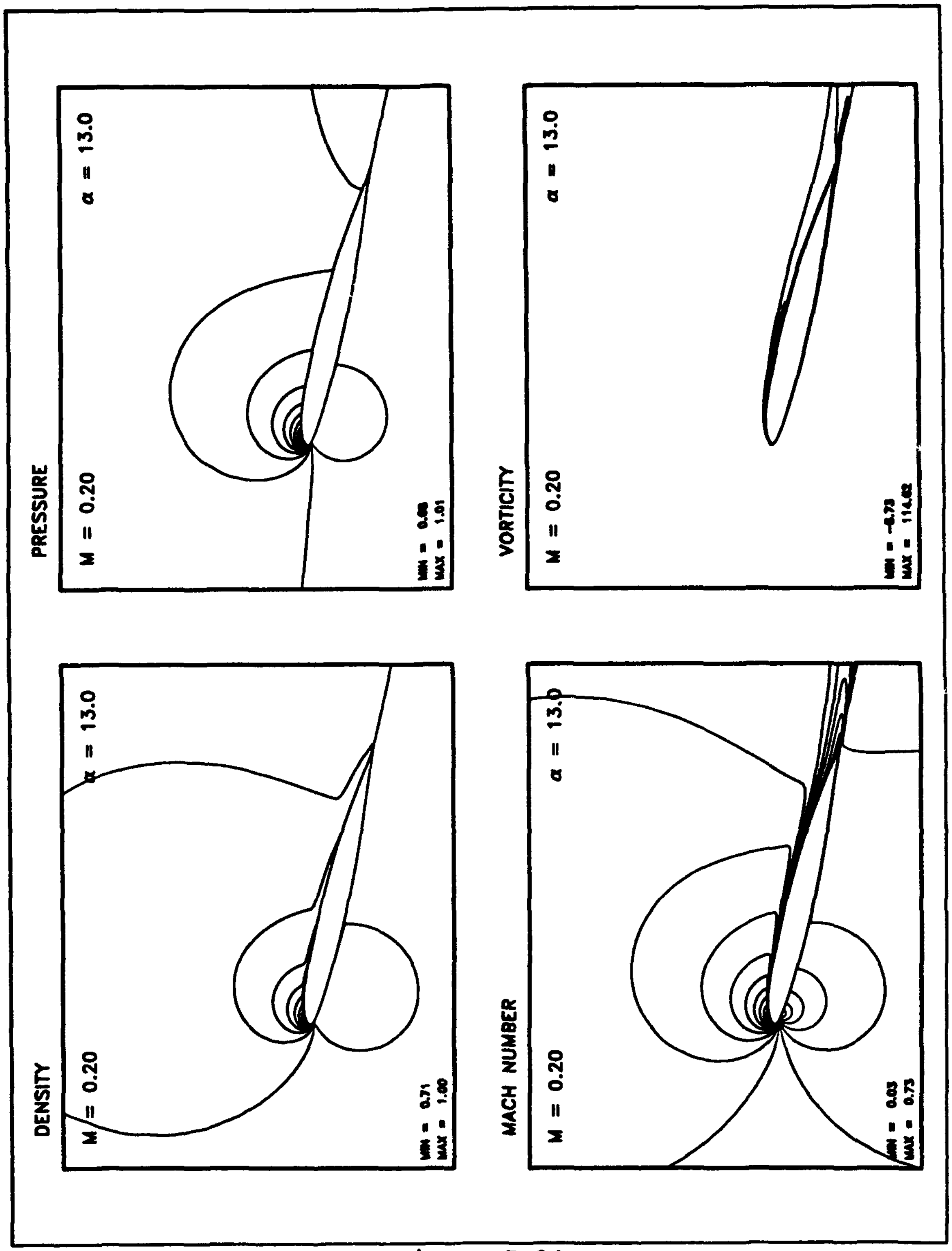


Figure 5.24
Sikorsky SSC-A09
Steady State (M=0.2)

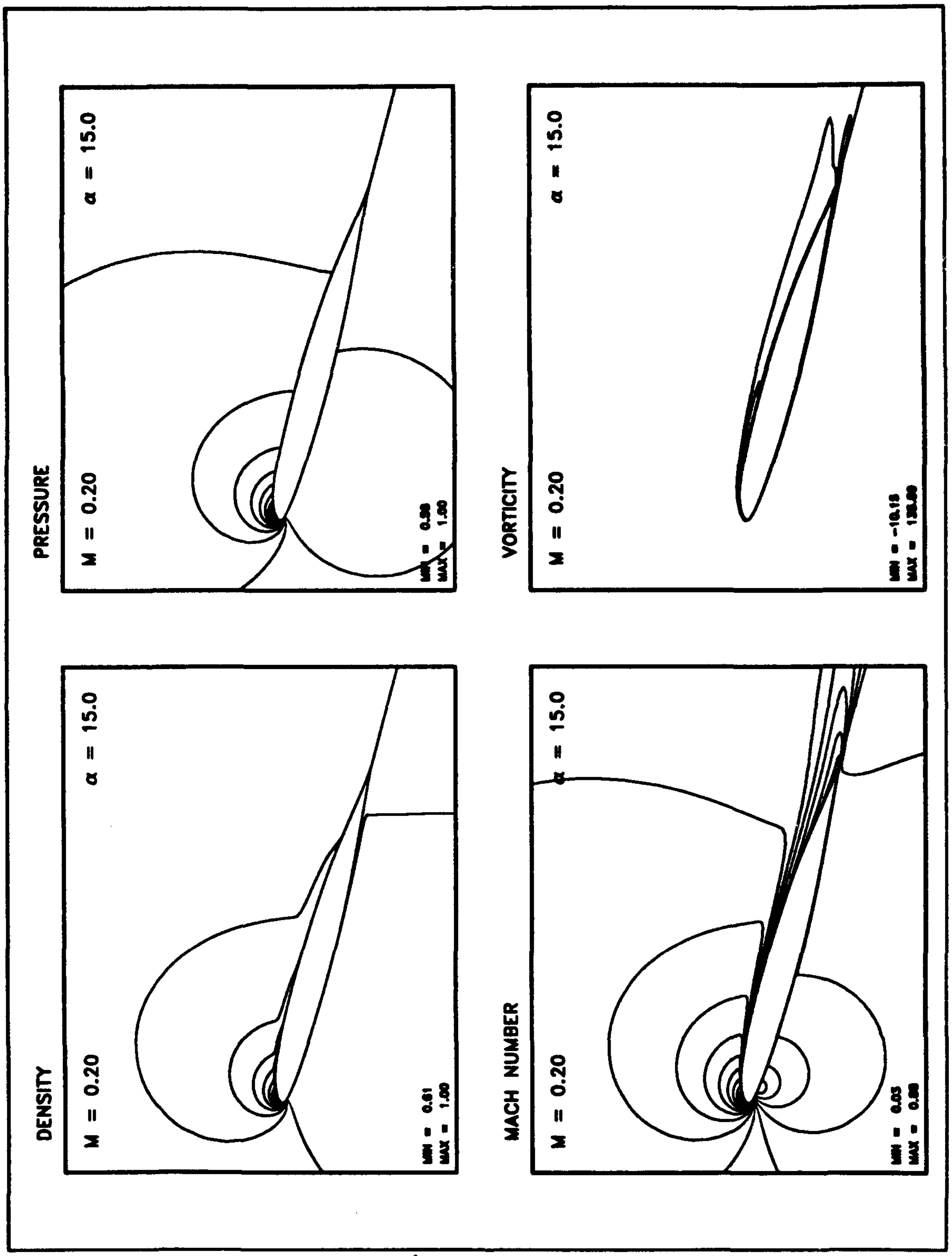


Figure 5.25
Sikorsky SSC-A09
Steady State (M=0.2)

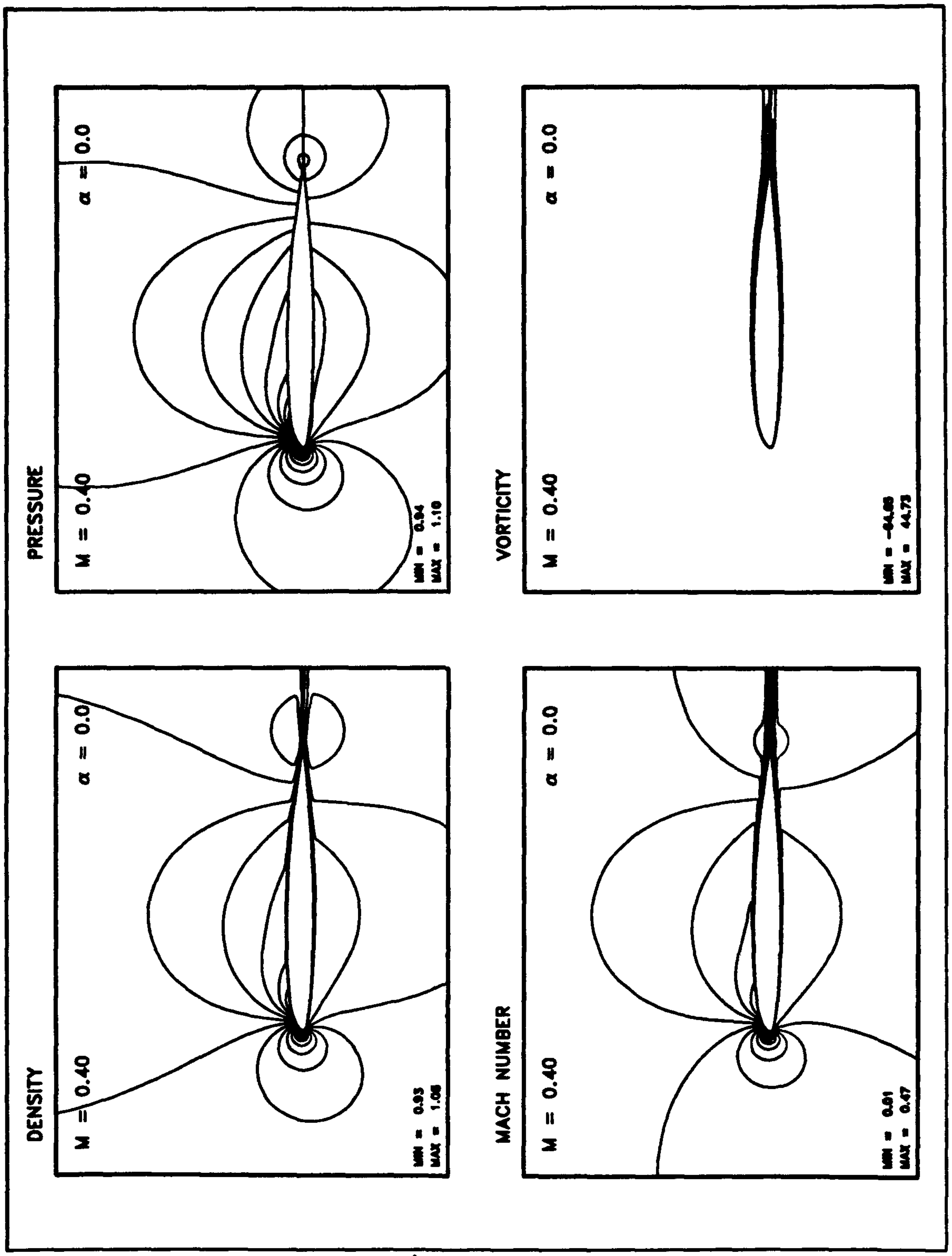


Figure 5.26
Sikorsky SSC-A09
Steady State (M=0.4)

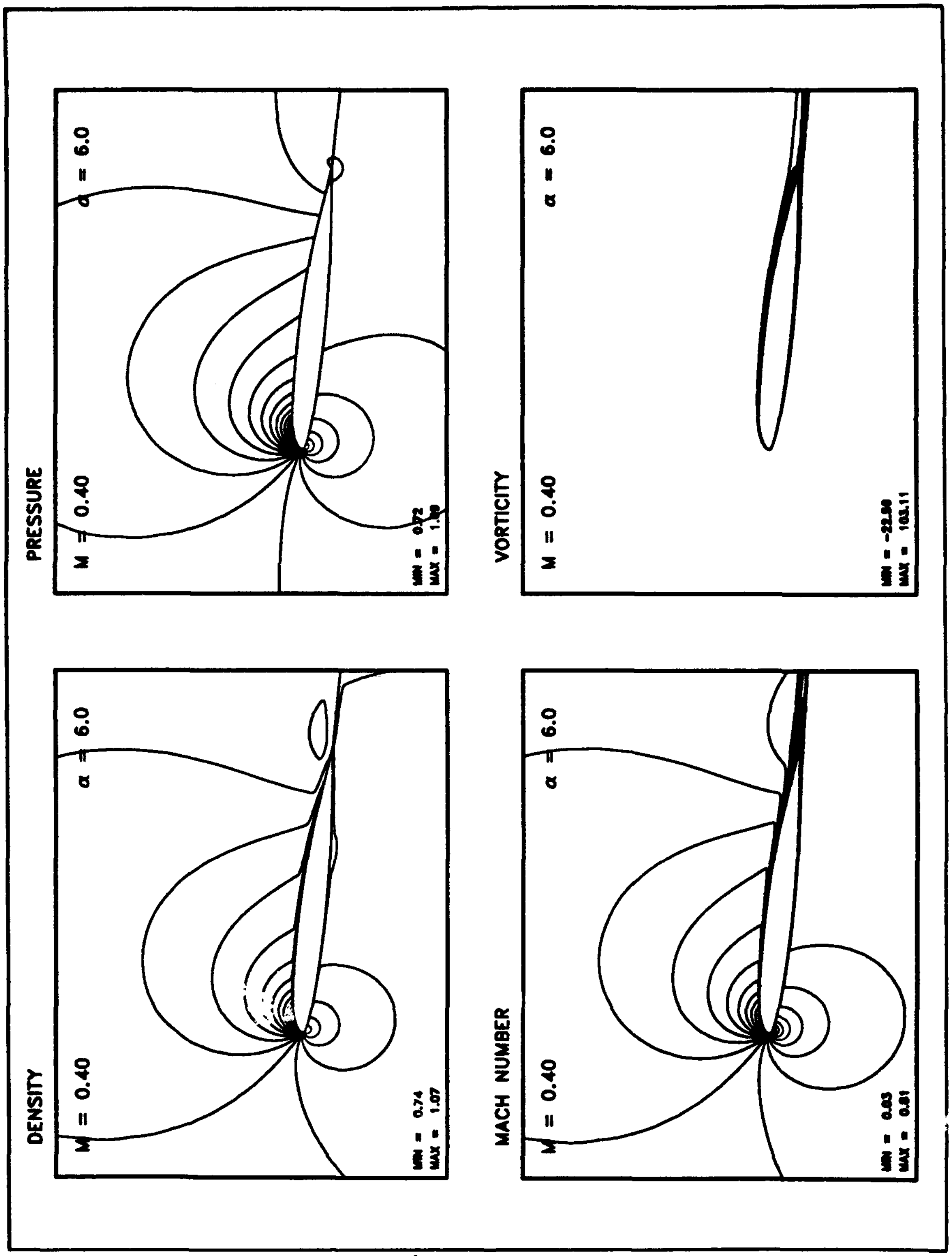


Figure 5.27
Sikorsky SSC-A09
Steady State (M=0.4)

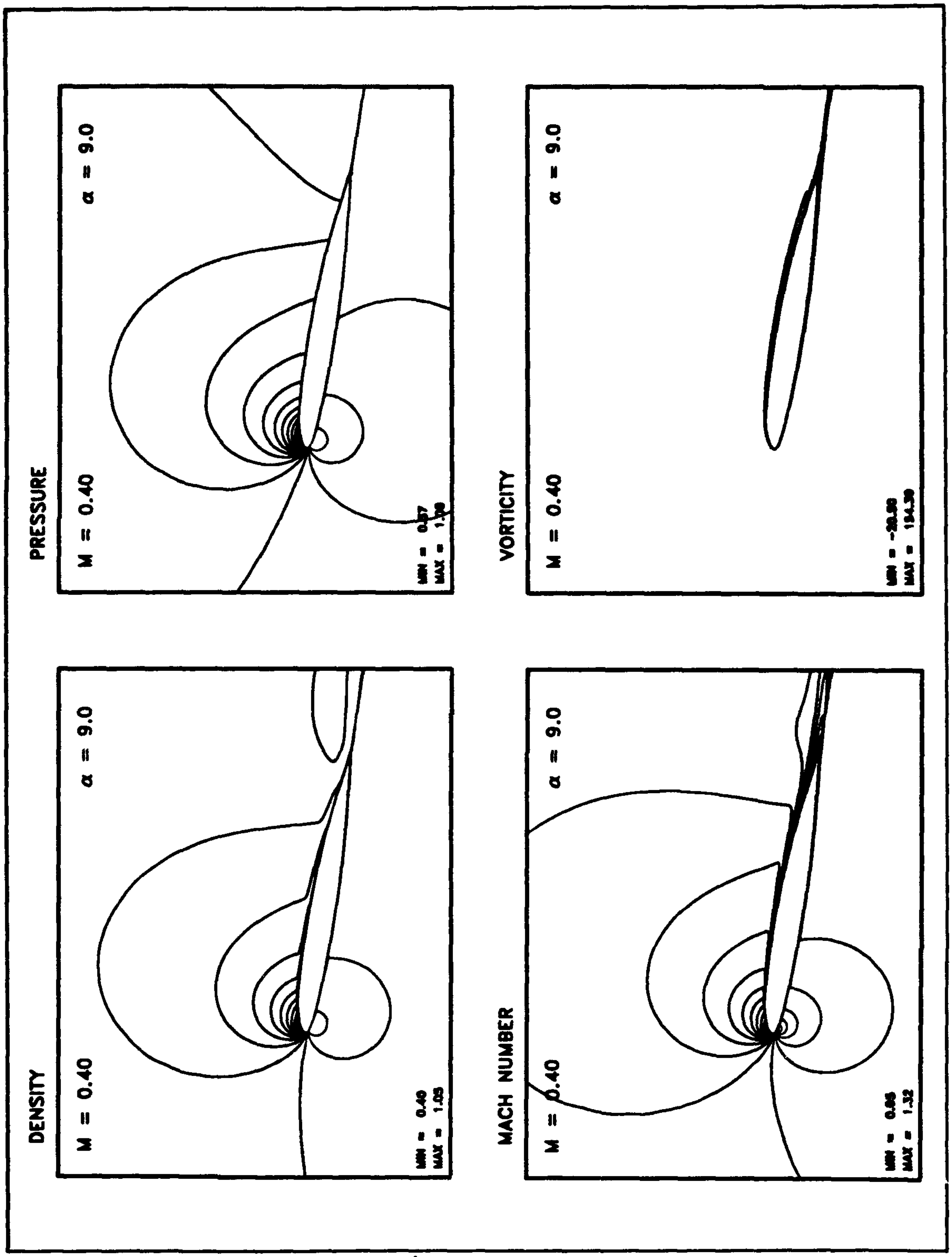


Figure 5.28
Sikorsky SSC-A09
Steady State (M=0.4)

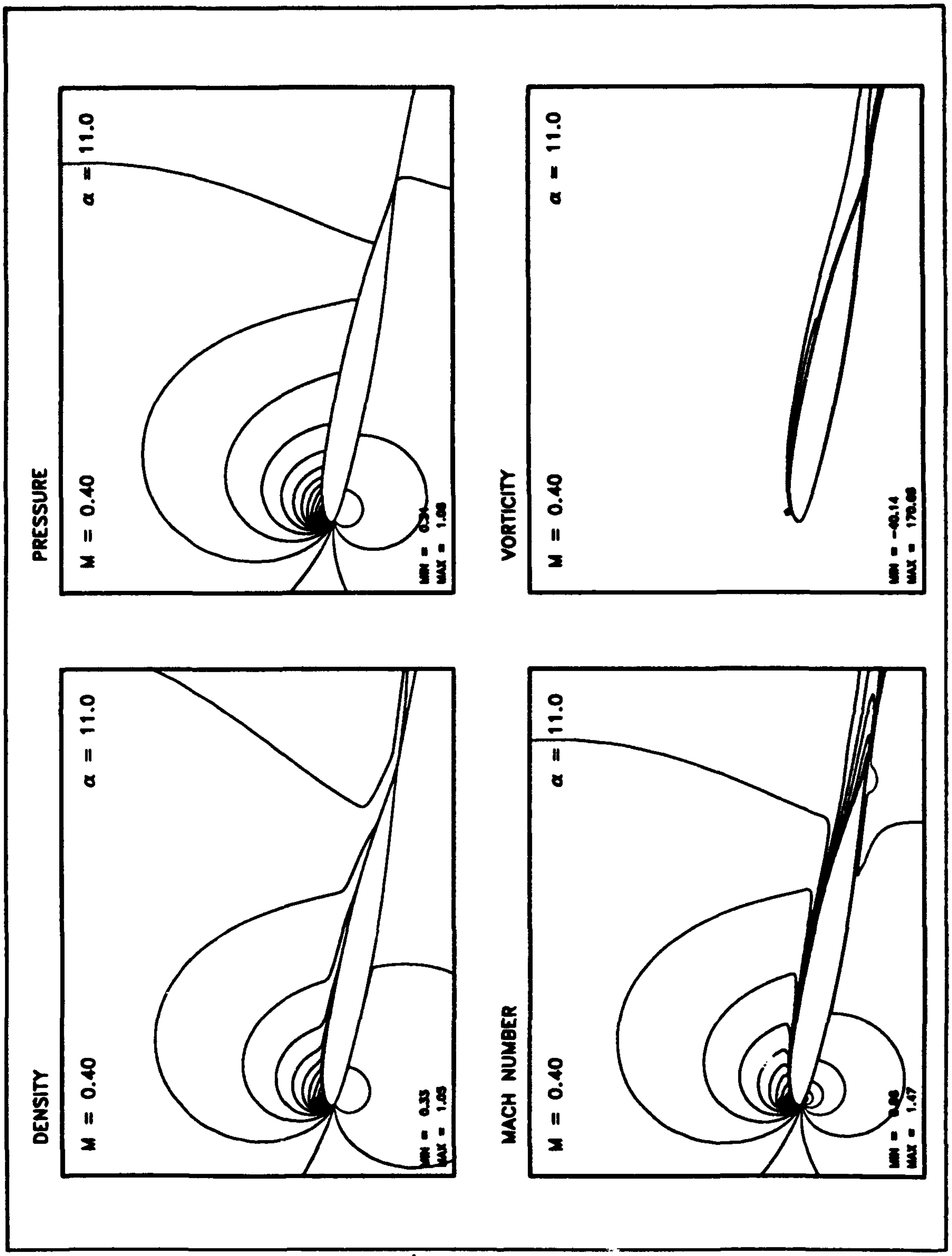


Figure 5.29
Sikorsky SSC-A09
Steady State (M=0.4)

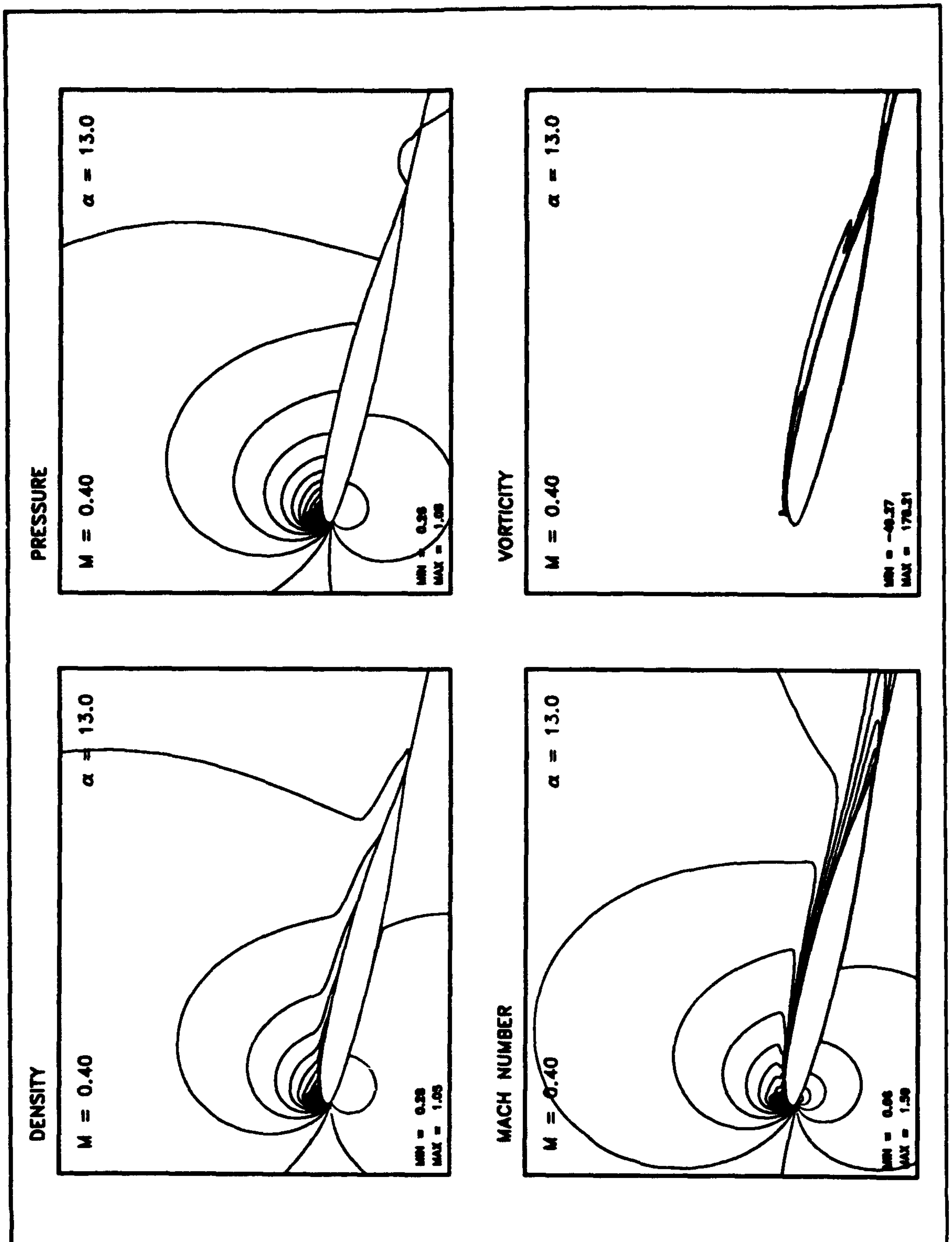


Figure 5.30
Sikorsky SSC-A09
Steady State (M=0.4)

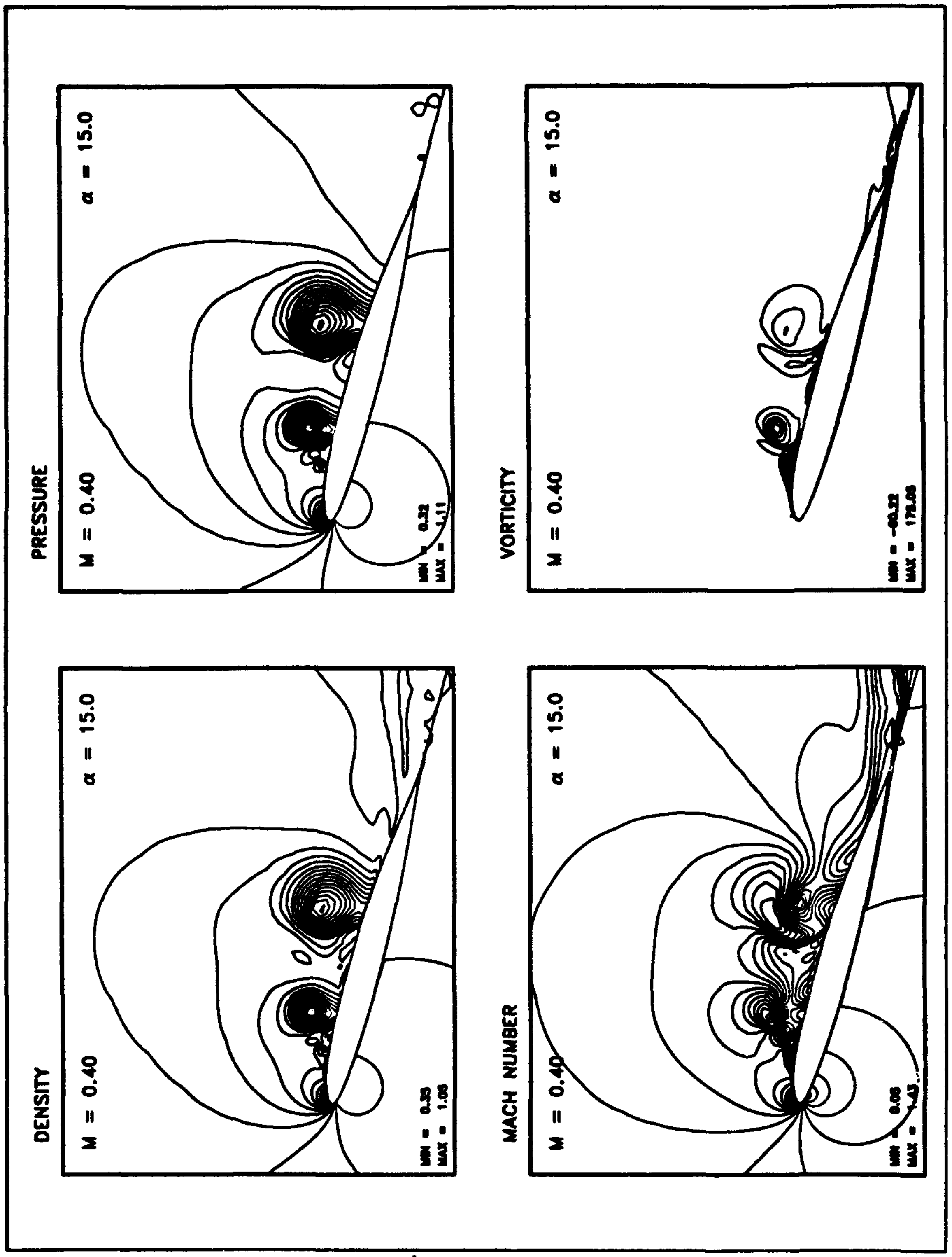


Figure 5.31
Sikorsky SSC-A09
Steady State (M=0.4)

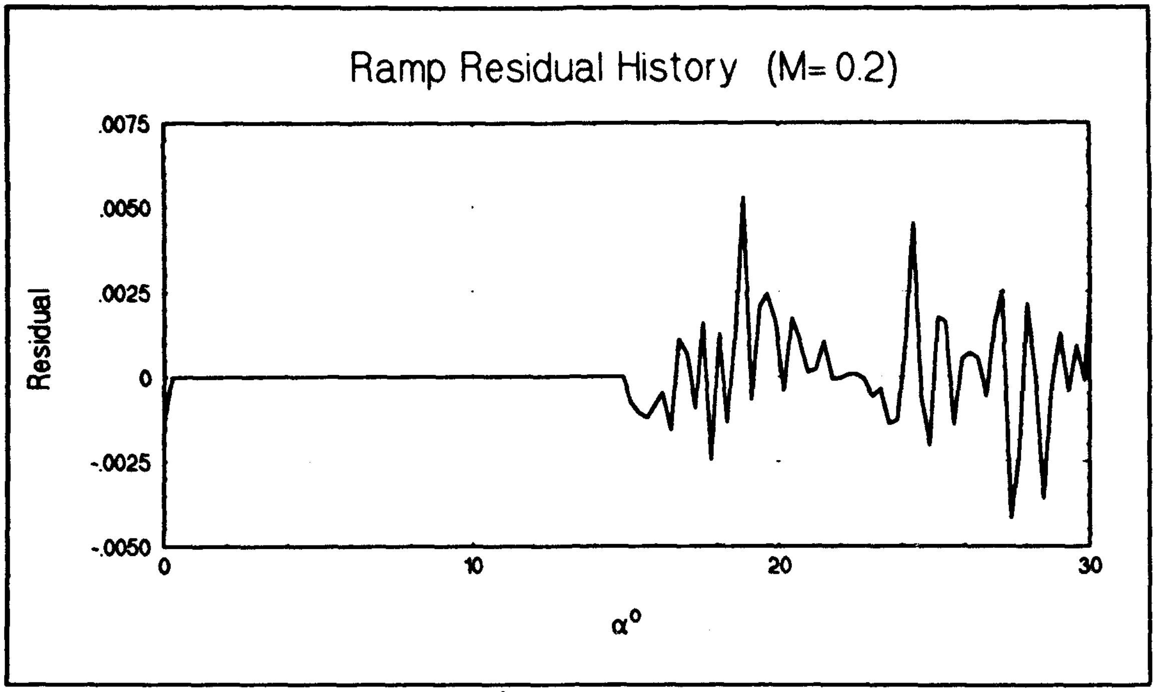


Figure 5.32
Ramp Residual History (M=0.2)

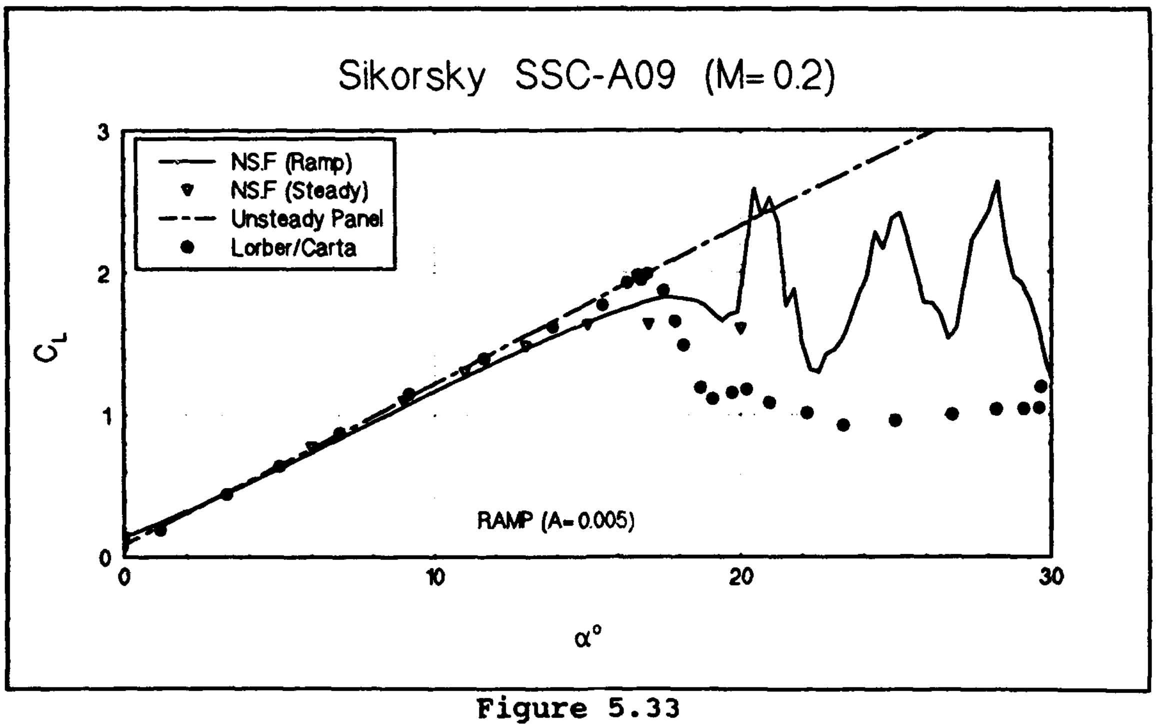


Figure 5.33
Ramp C_{L_s} (M=0.2)

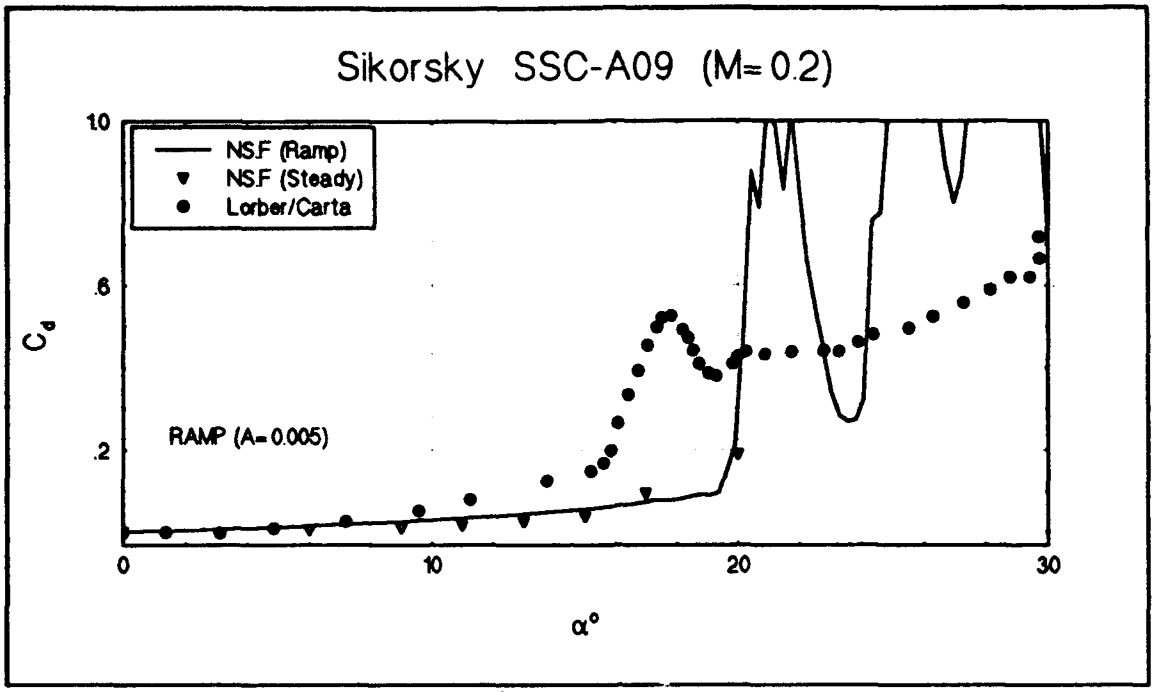


Figure 5.34
Ramp C_{ds} (M=0.2)

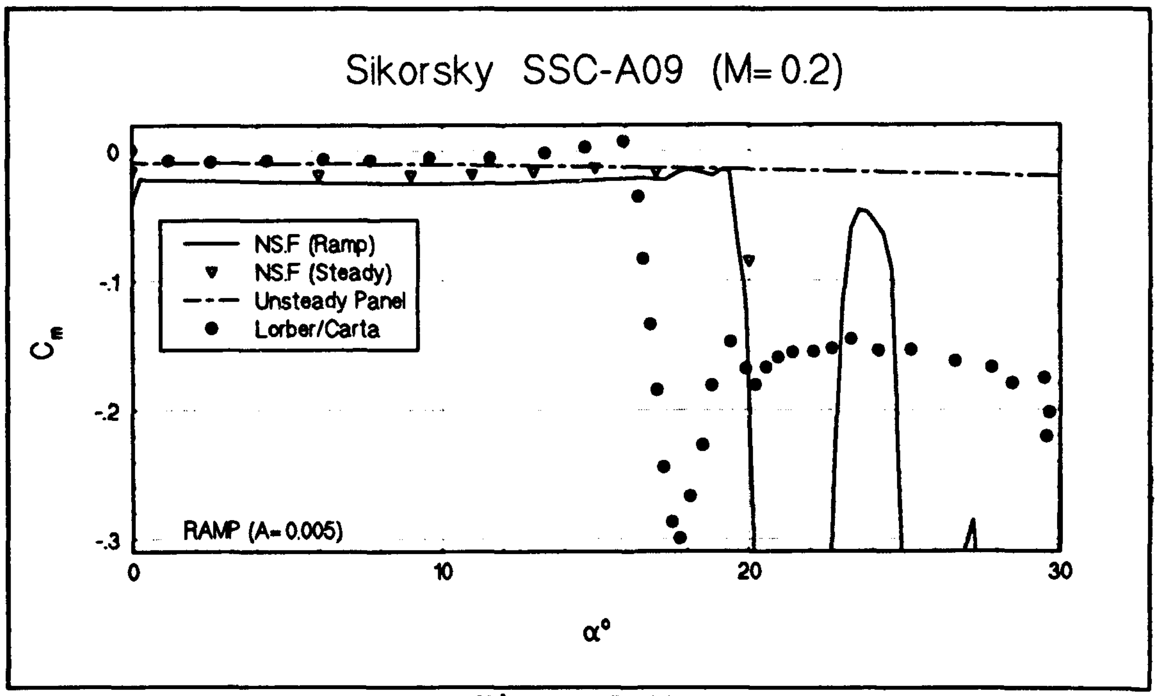


Figure 5.35
Ramp C_{Ms} (M=0.2)

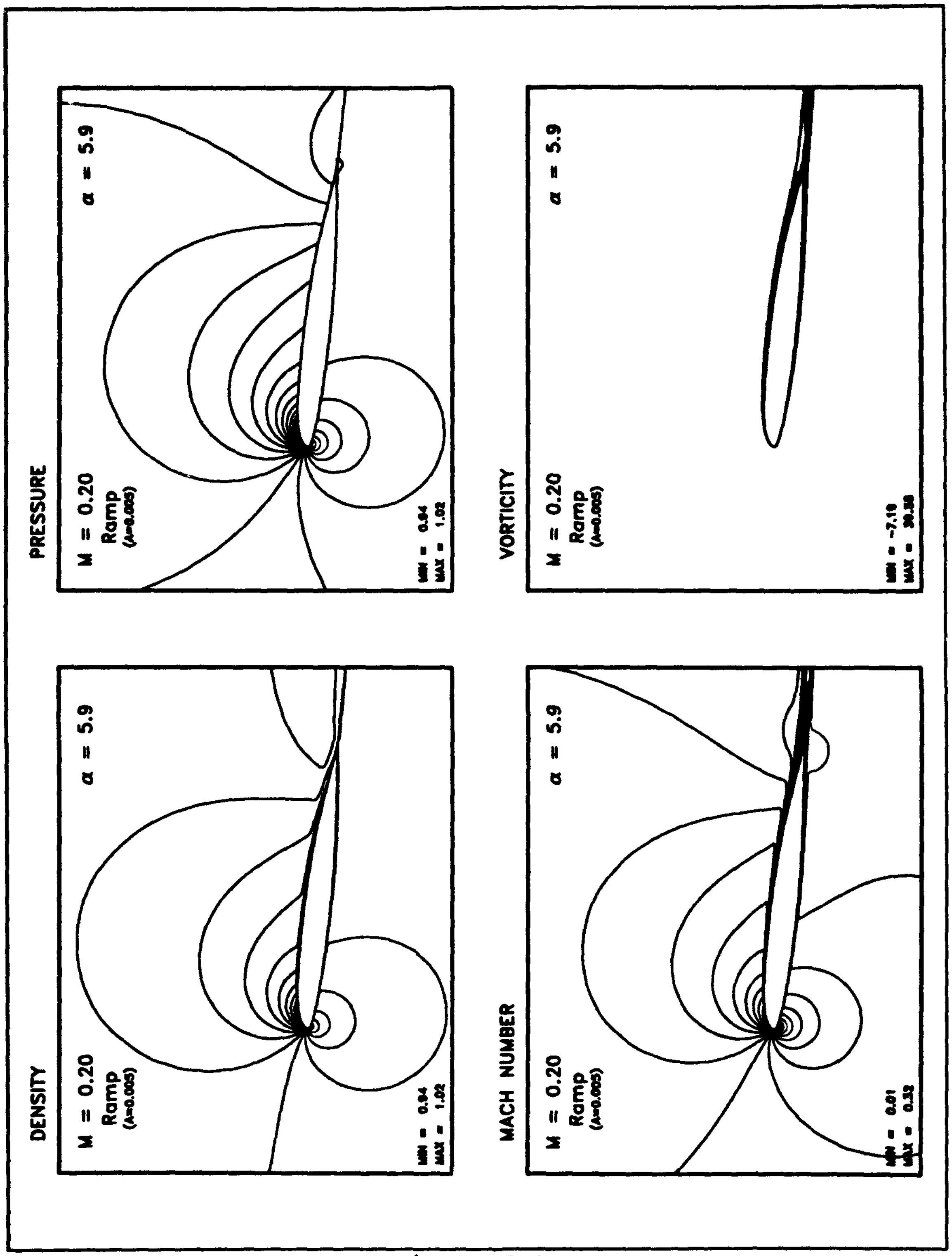


Figure 5.36
Sikorsky SSC-A09
Ramp (A=0.005, M=0.2)

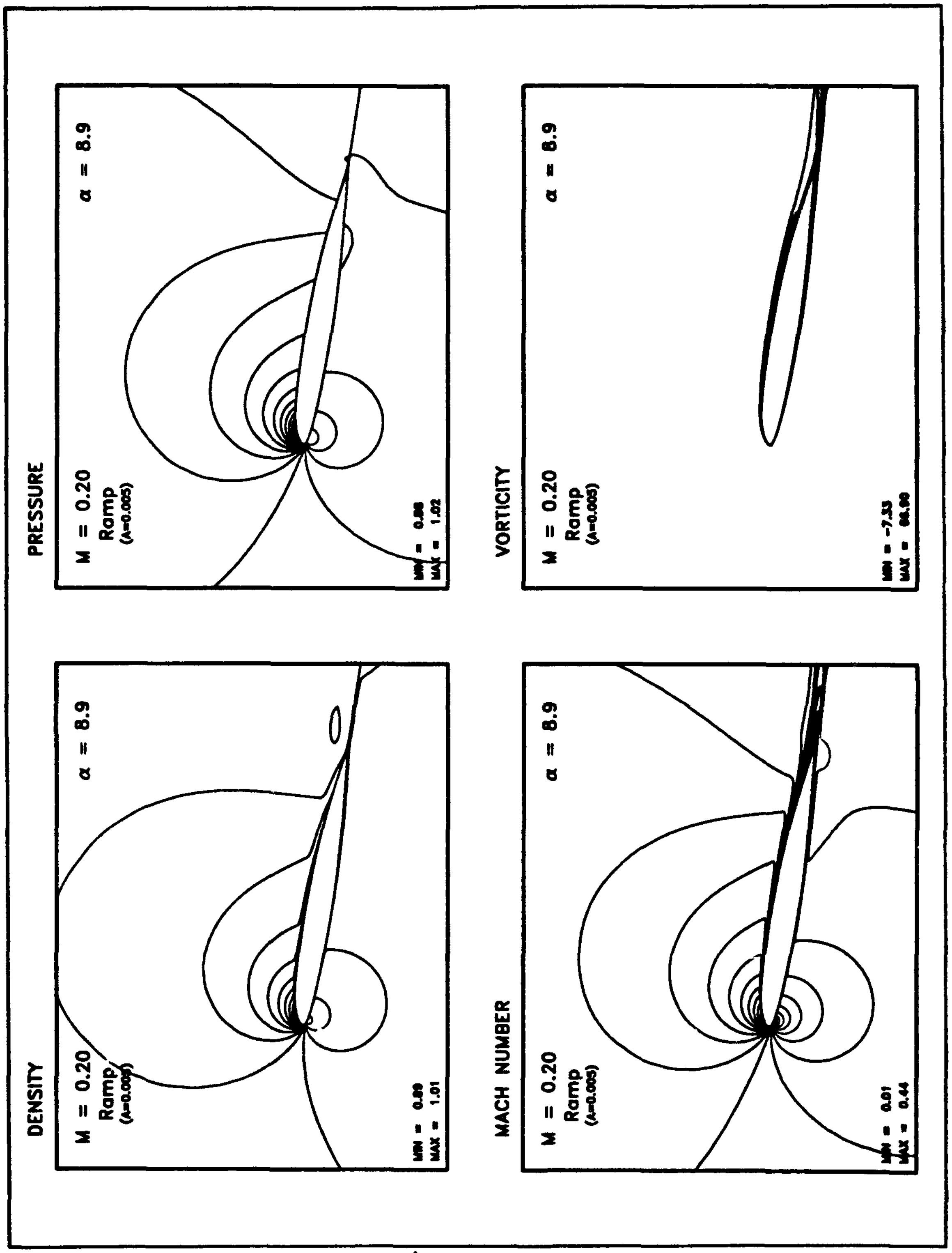


Figure 5.37
Sikorsky SSC-A09
Ramp (A=0.005, M=0.2)

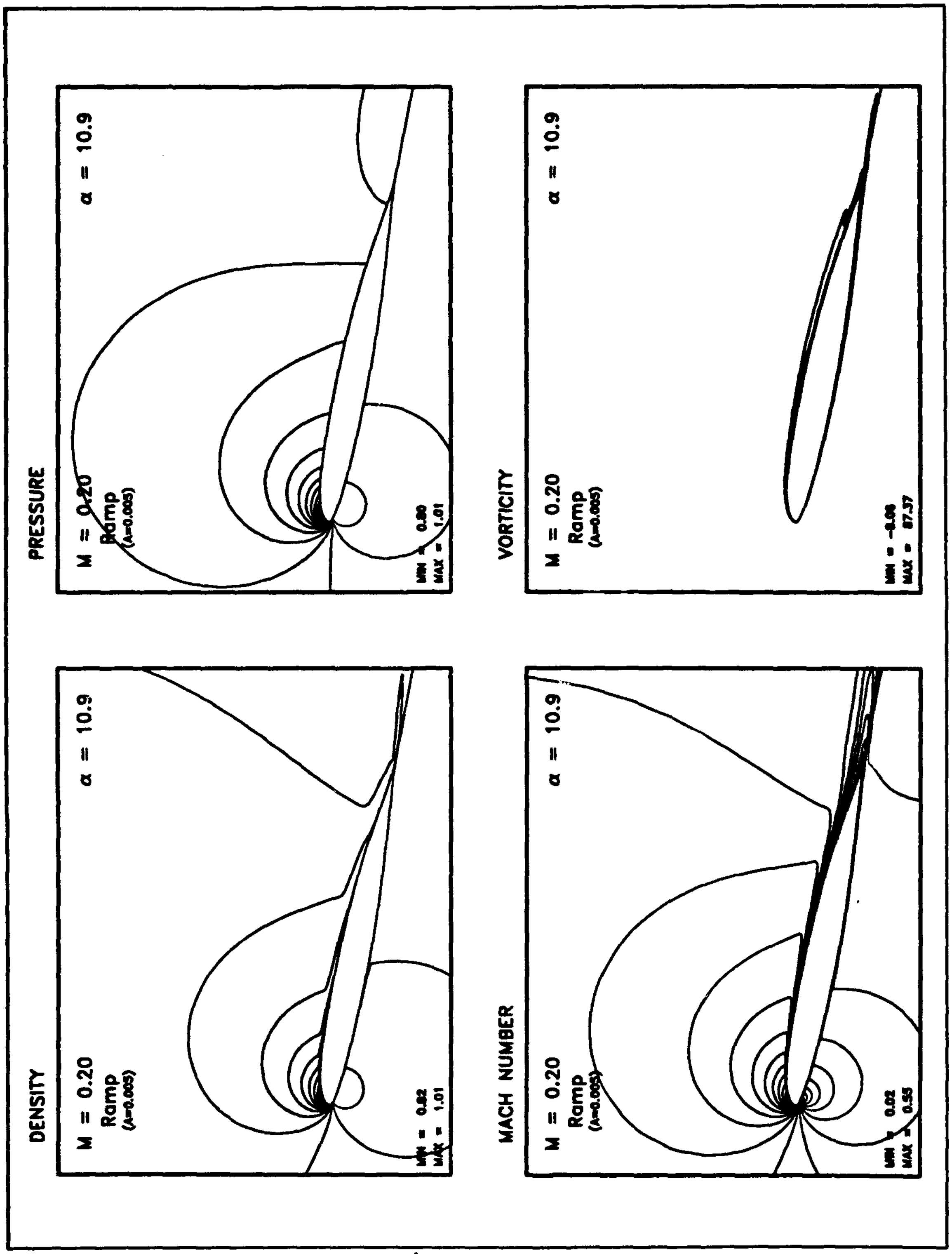


Figure 5.38
Sikorsky SSC-A09
Ramp (A=0.005, M=0.2)

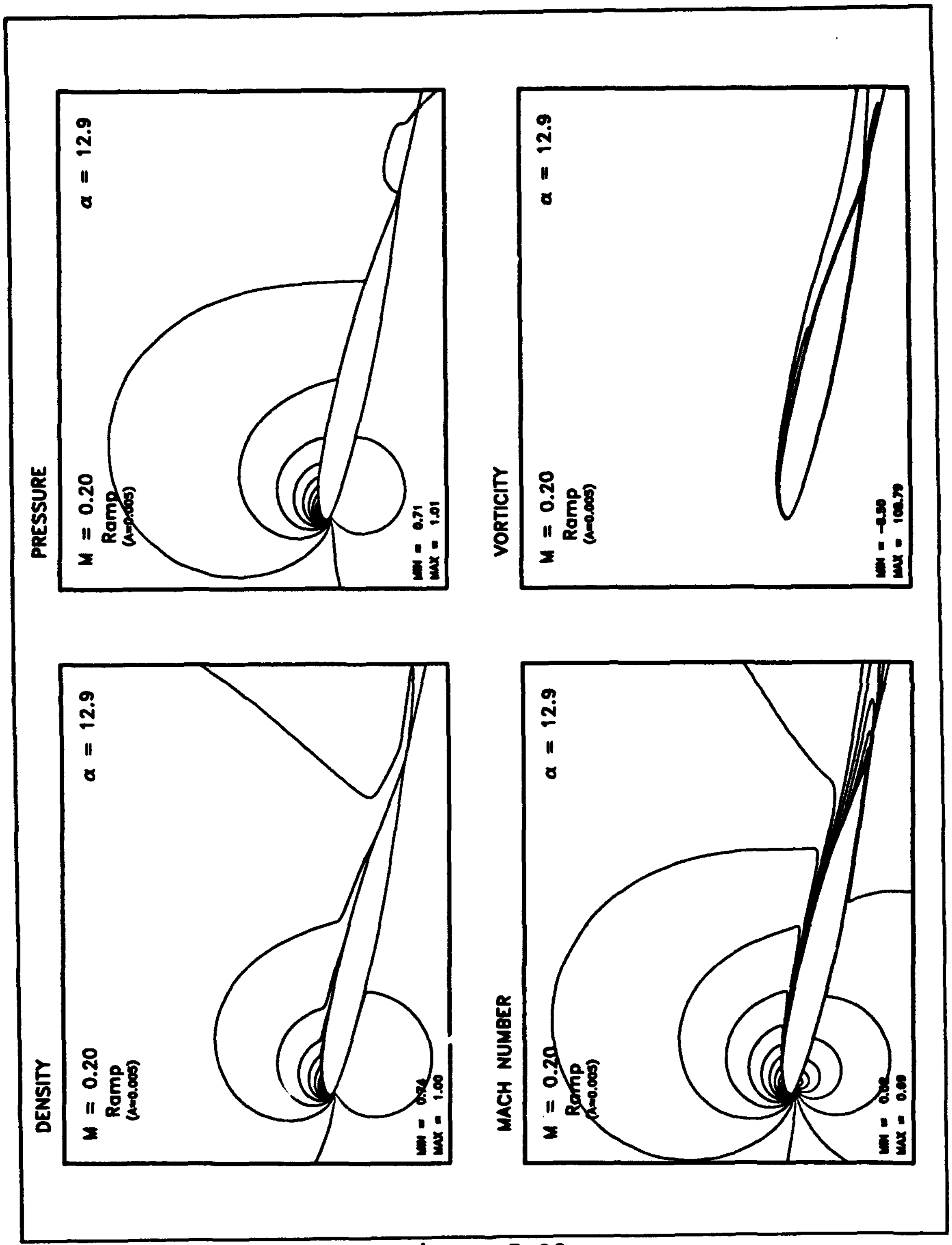


Figure 5.39
Sikorsky SSC-A09
Ramp (A=0.005, M=0.2)

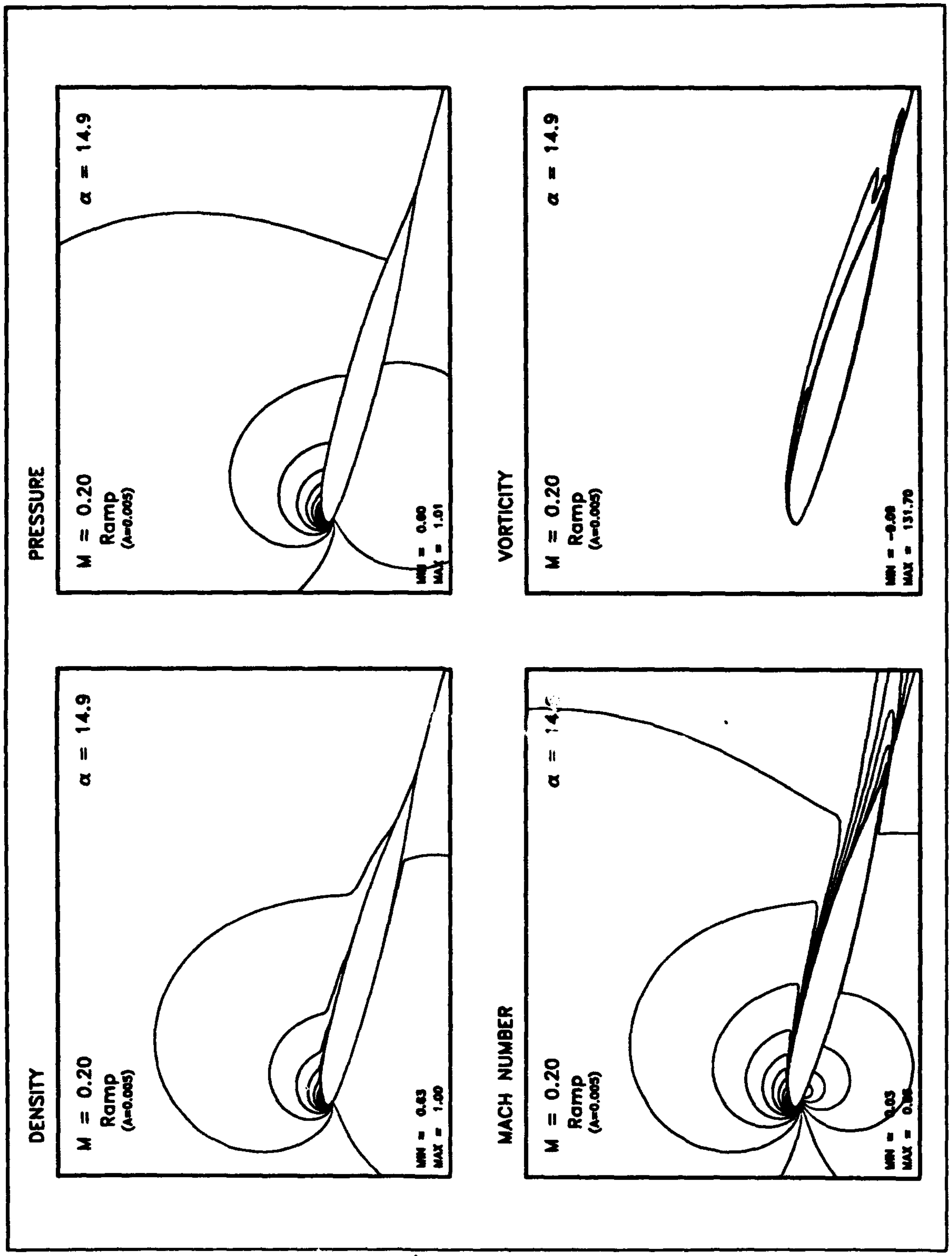


Figure 5.40
Sikorsky SSC-A09
Ramp (A=0.005, M=0.2)

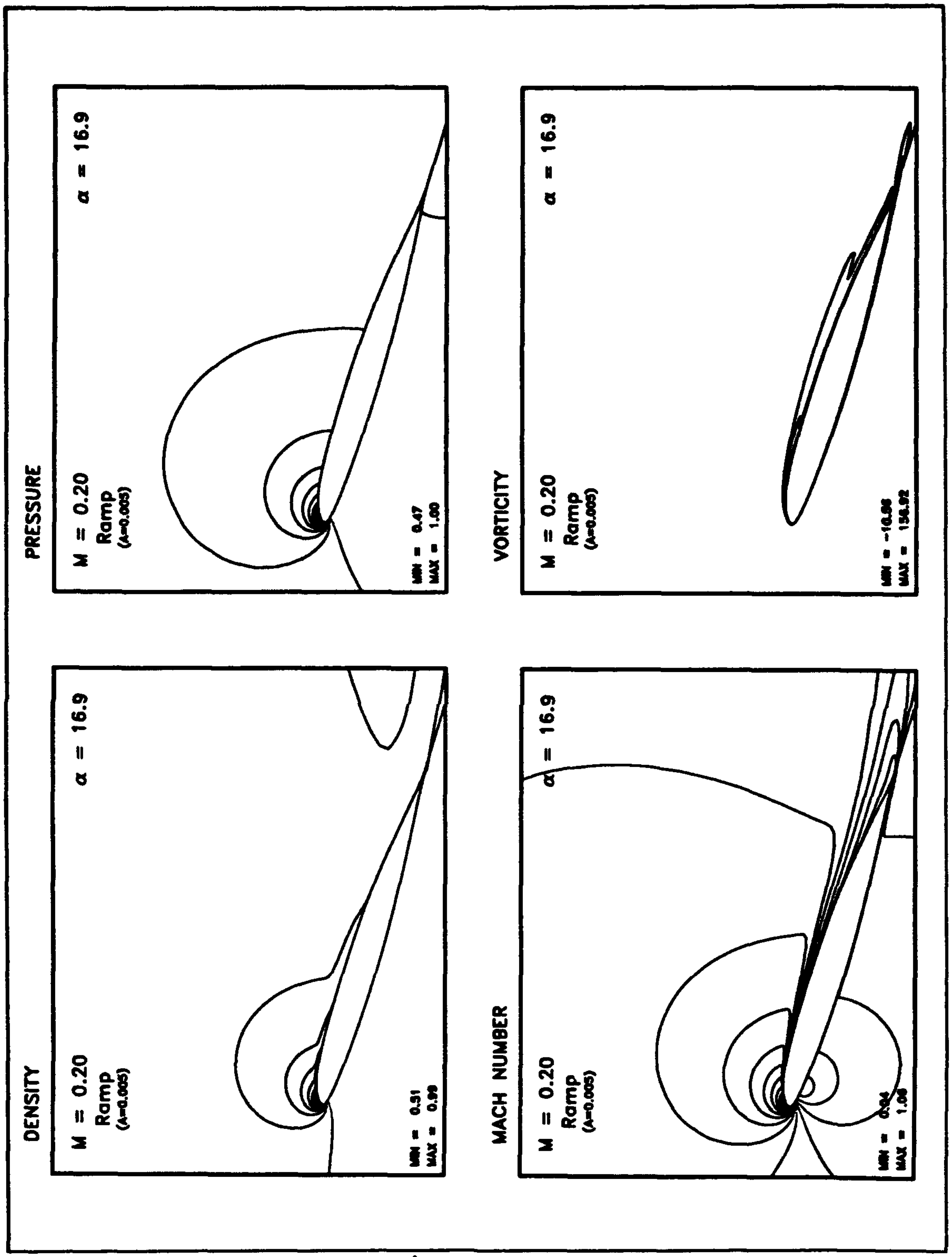


Figure 5.41
Sikorsky SSC-A09
Ramp (A=0.005, M=0.2)

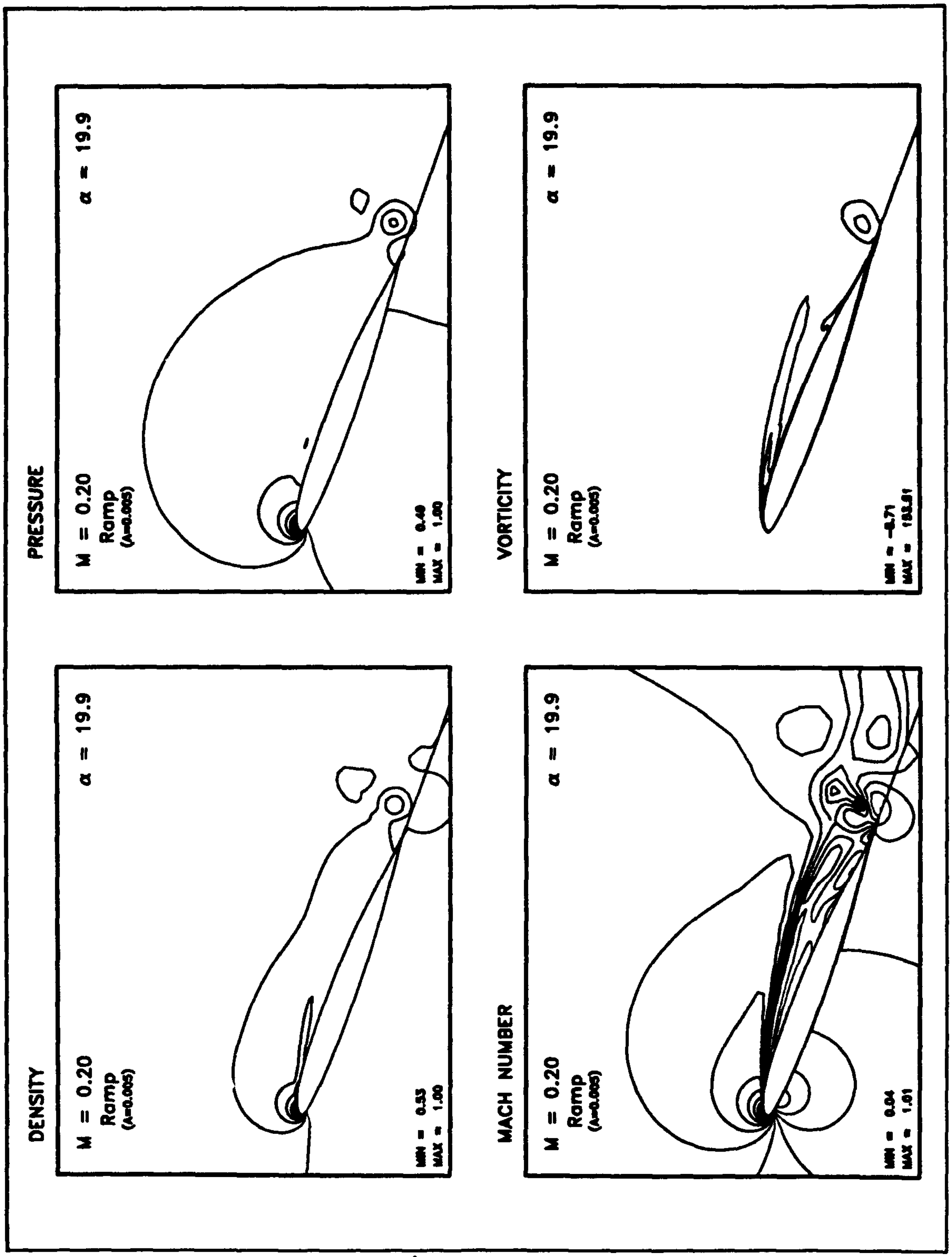


Figure 5.42
Sikorsky SSC-A09
Ramp (A=0.005, M=0.2)

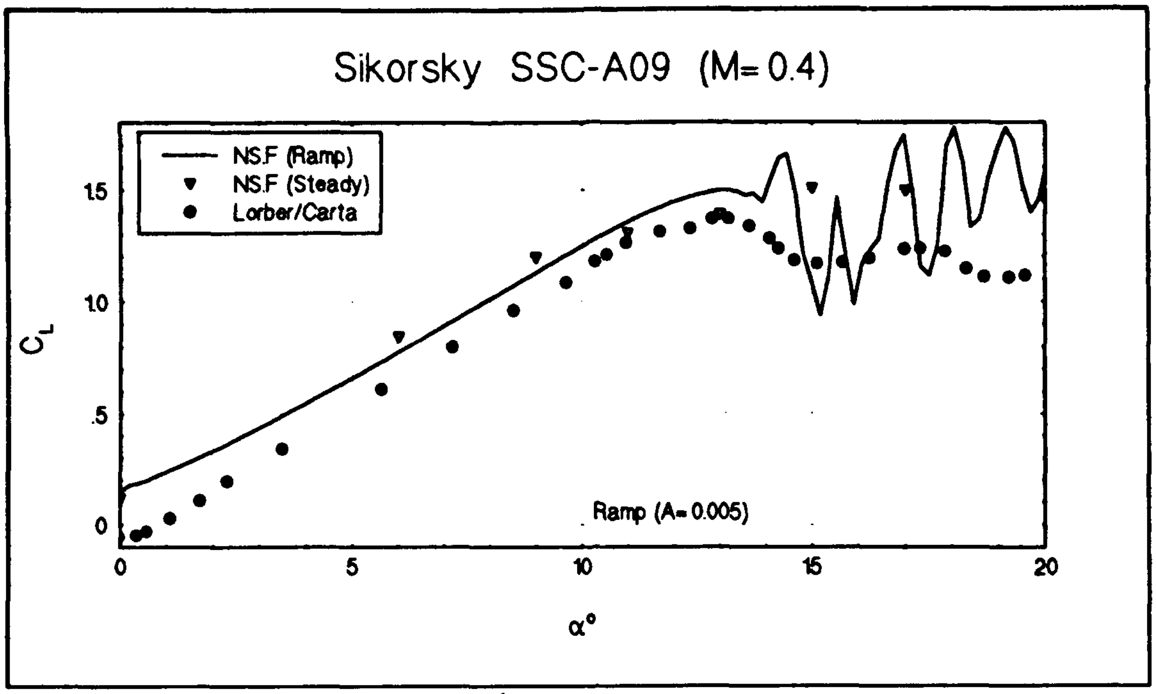


Figure 5.43
Ramp $C_{L\alpha}$ (M=0.4)

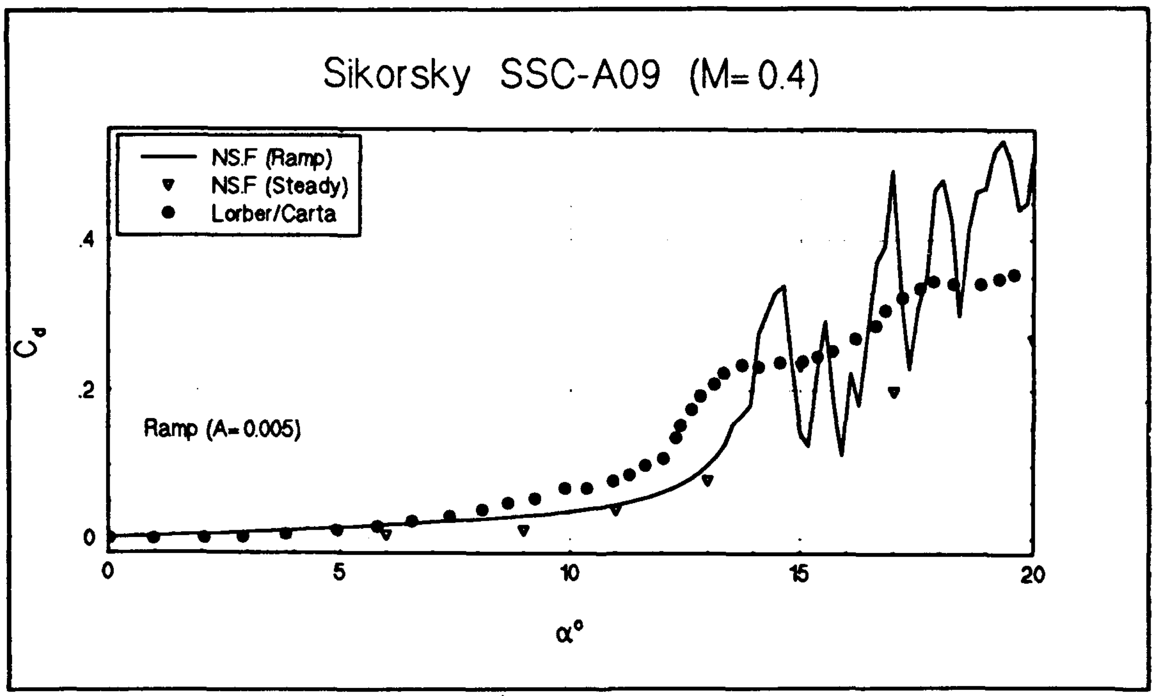


Figure 5.44
Ramp $C_{d\alpha}$ (M=0.4)

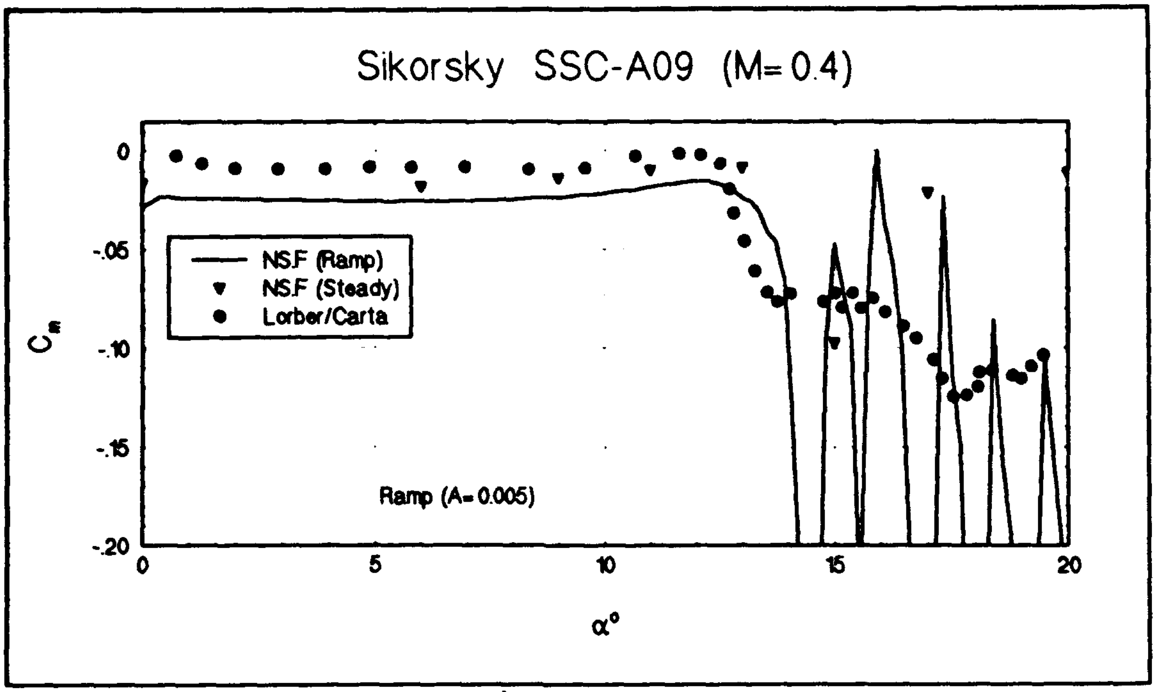


Figure 5.45
Ramp $C_{M\alpha}$ (M=0.4)

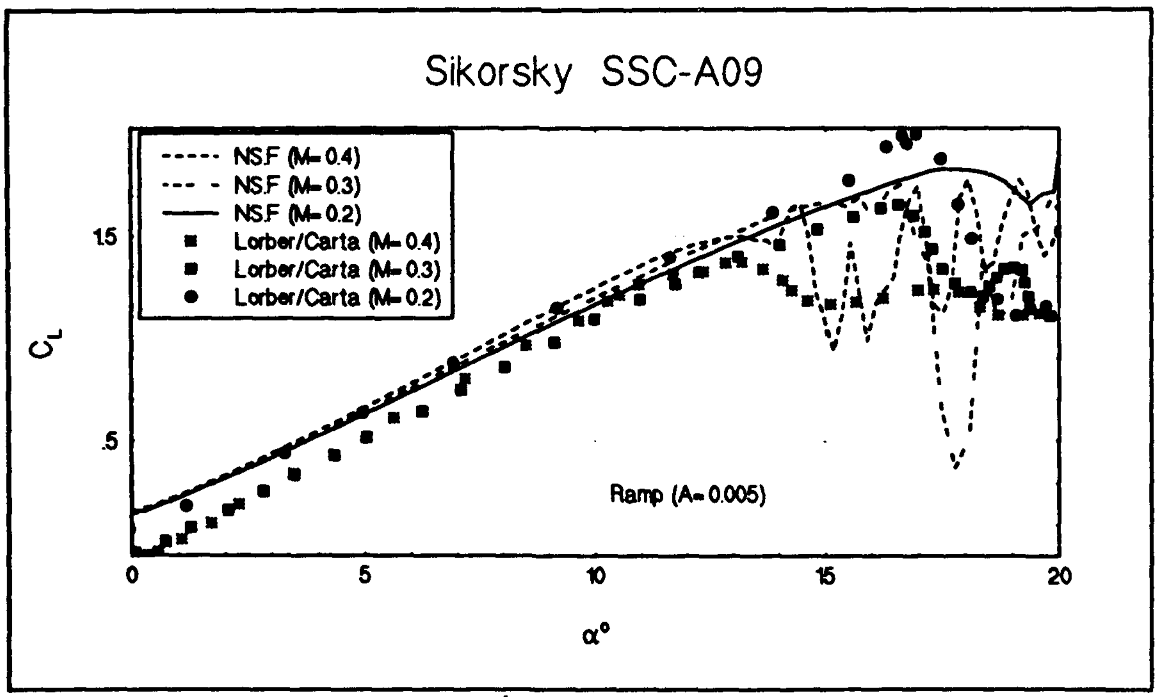


Figure 5.46
Ramp $C_{L\alpha}$ (M=0.2,0.3,0.4)

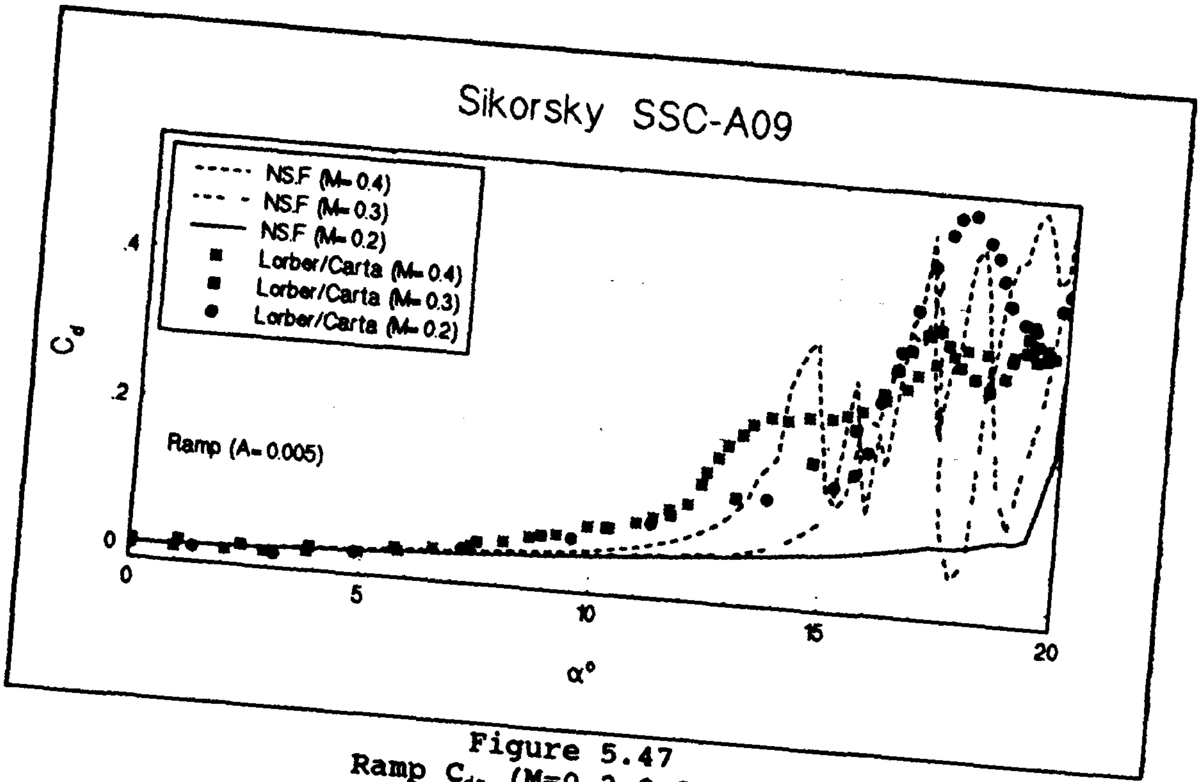


Figure 5.47
Ramp C_{ds} (M=0.2,0.3,0.4)

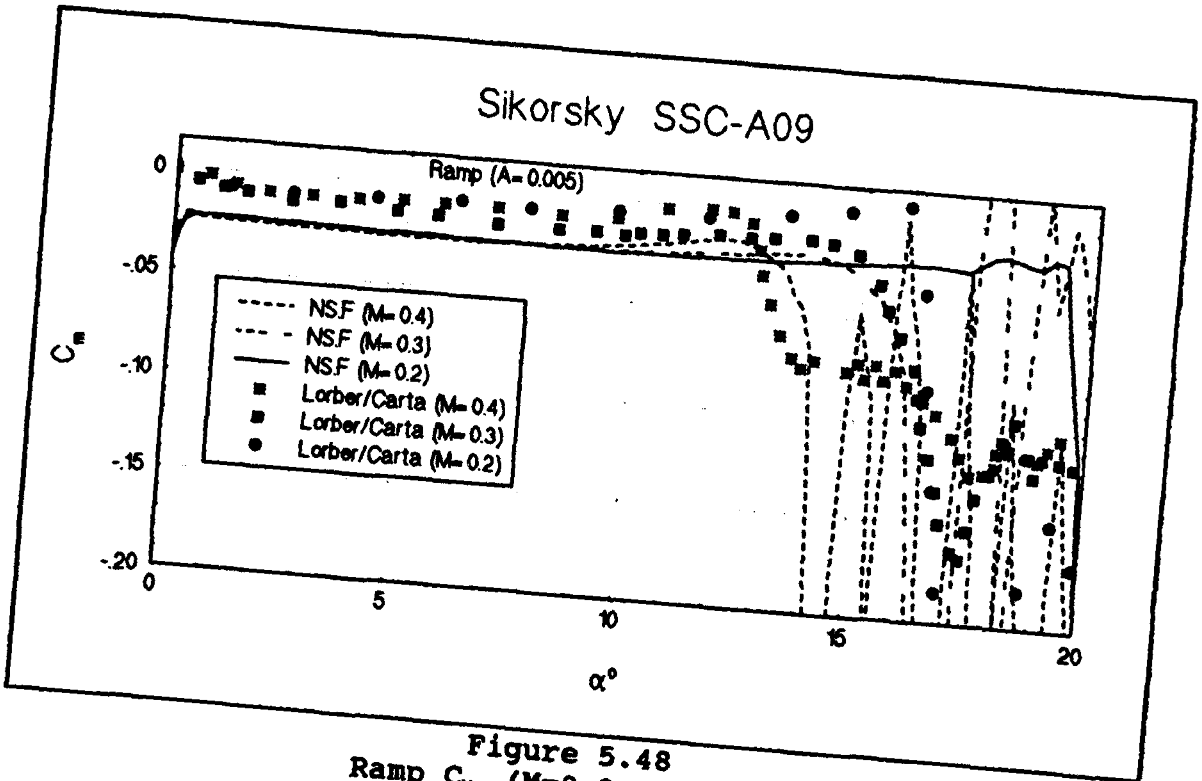


Figure 5.48
Ramp C_{Ms} (M=0.2,0.3,0.4)

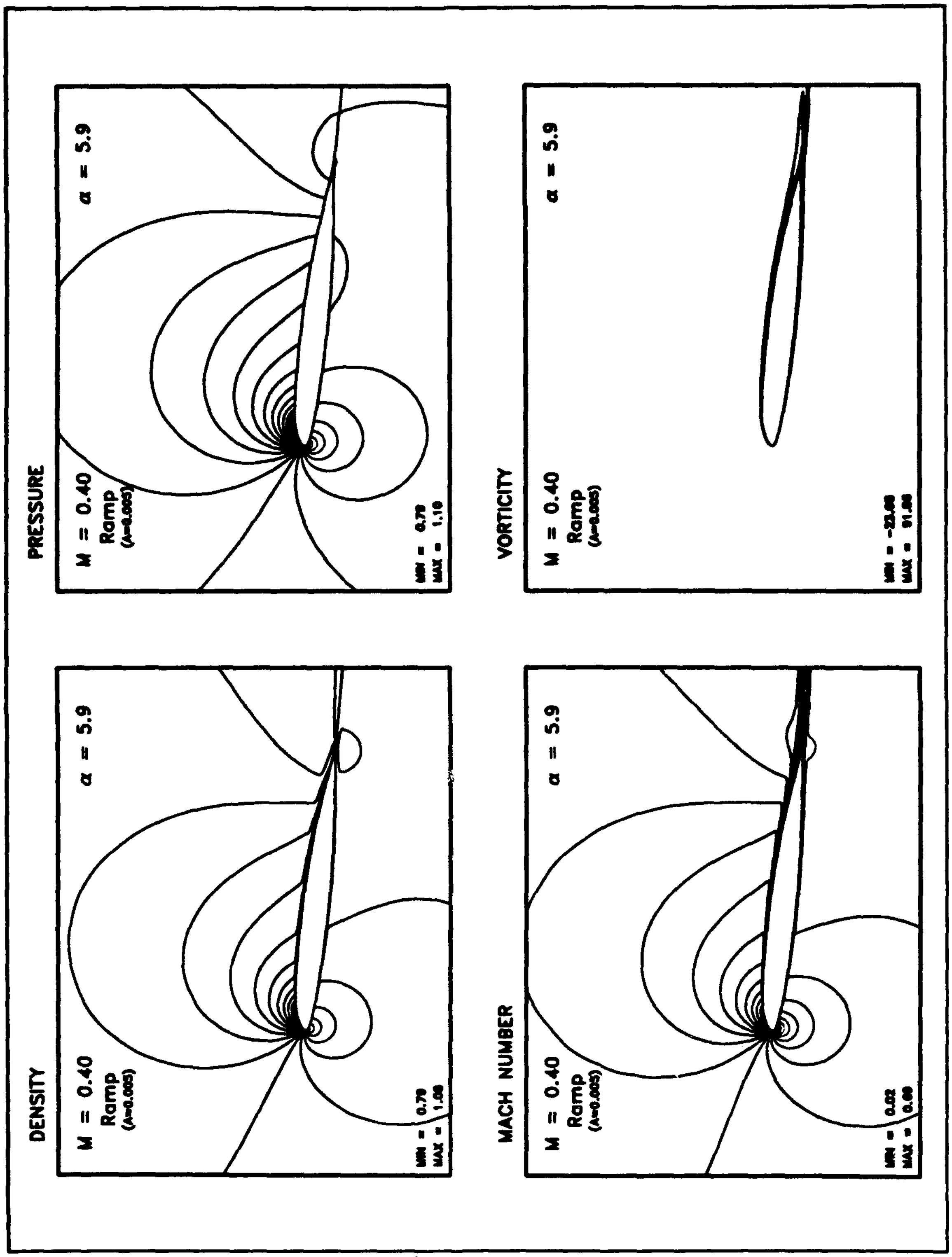


Figure 5.49
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

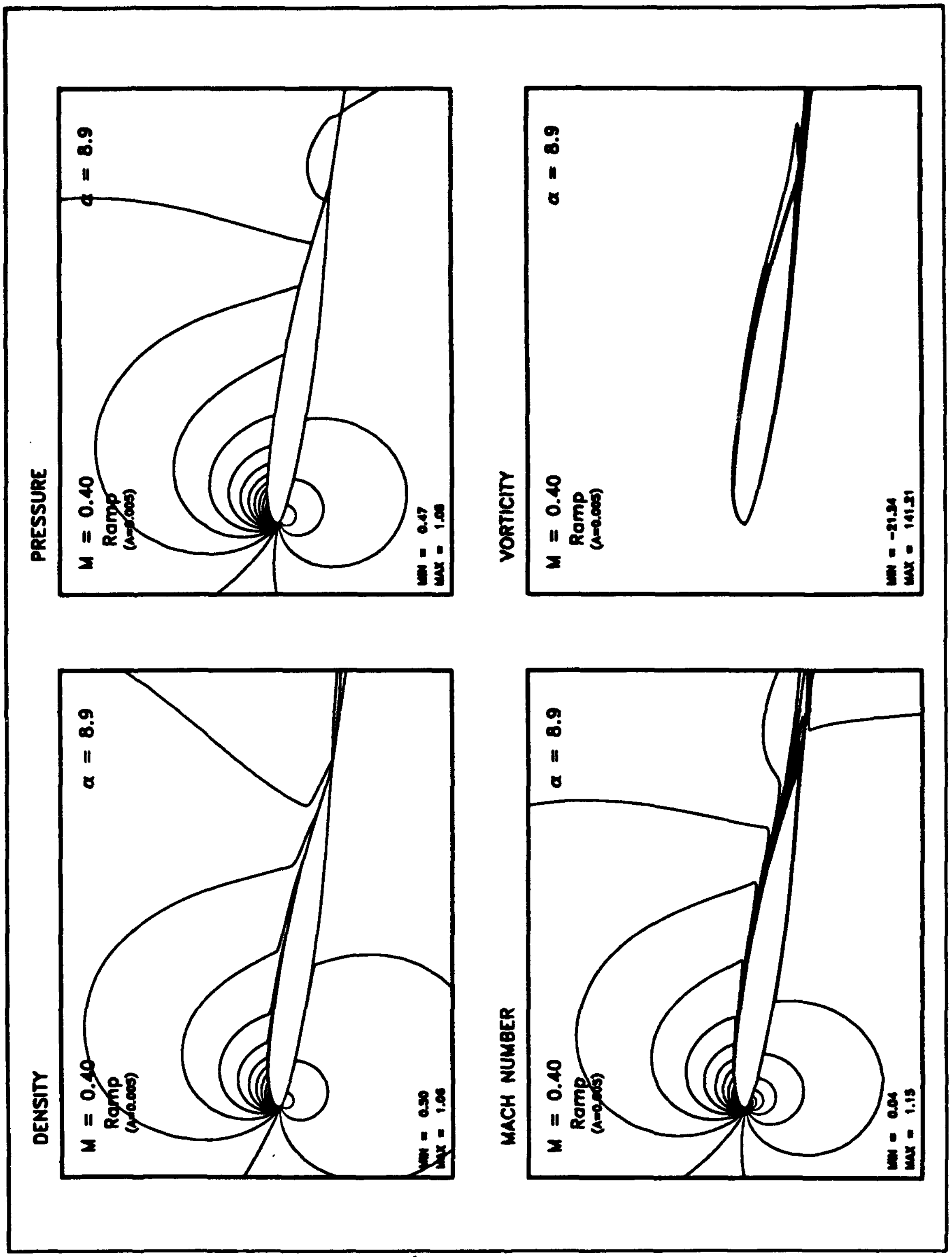


Figure 5.50
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

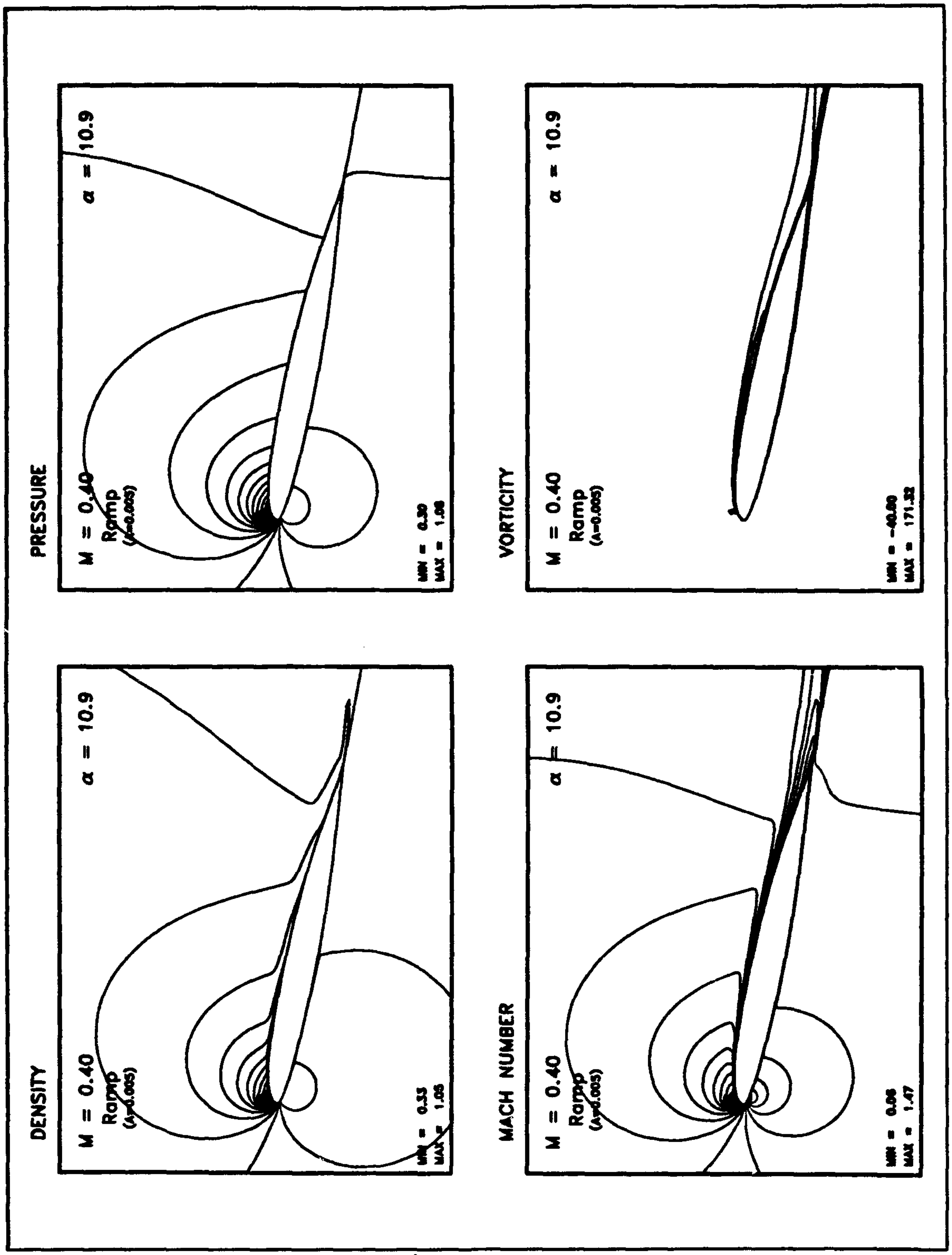


Figure 5.51
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

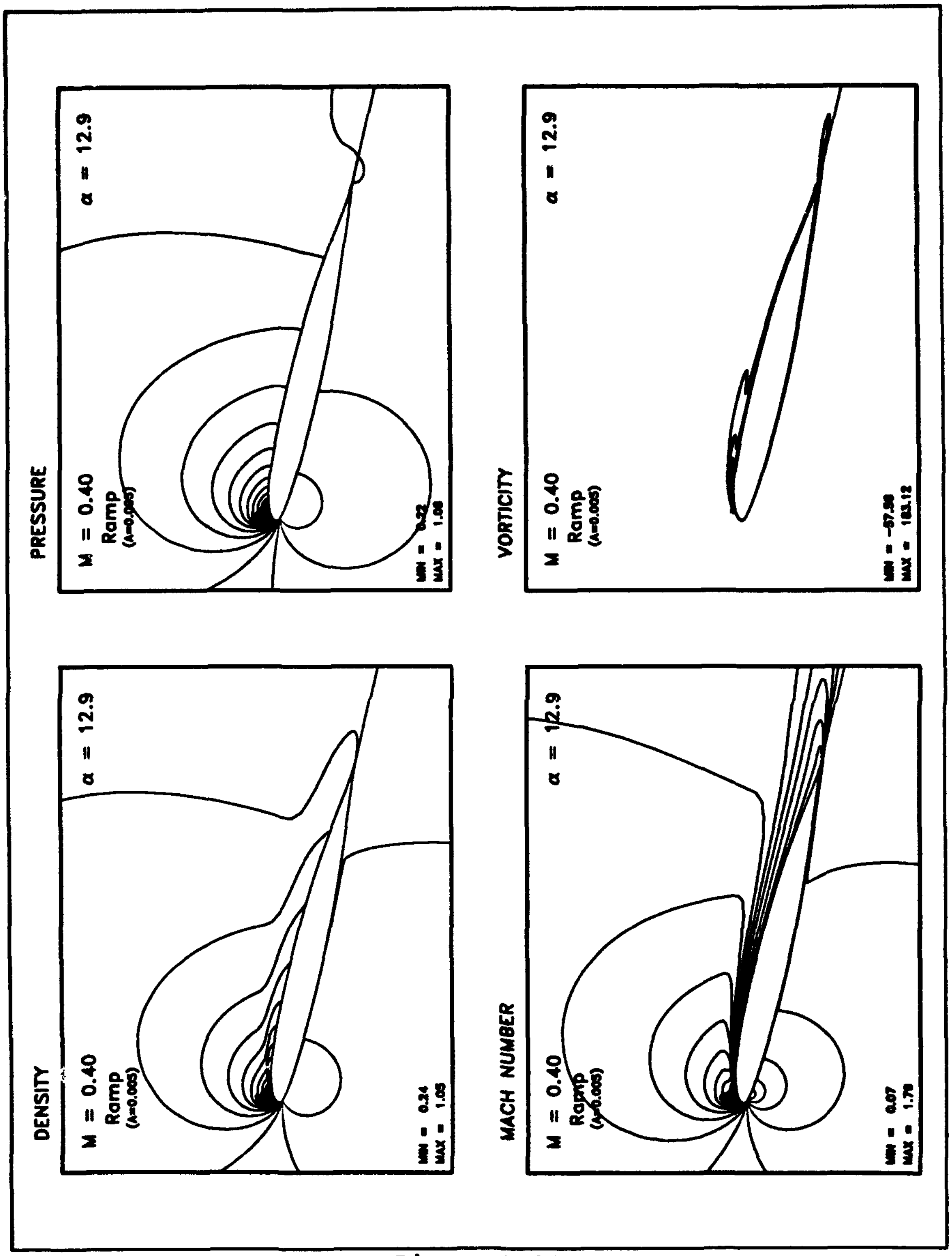


Figure 5.52
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

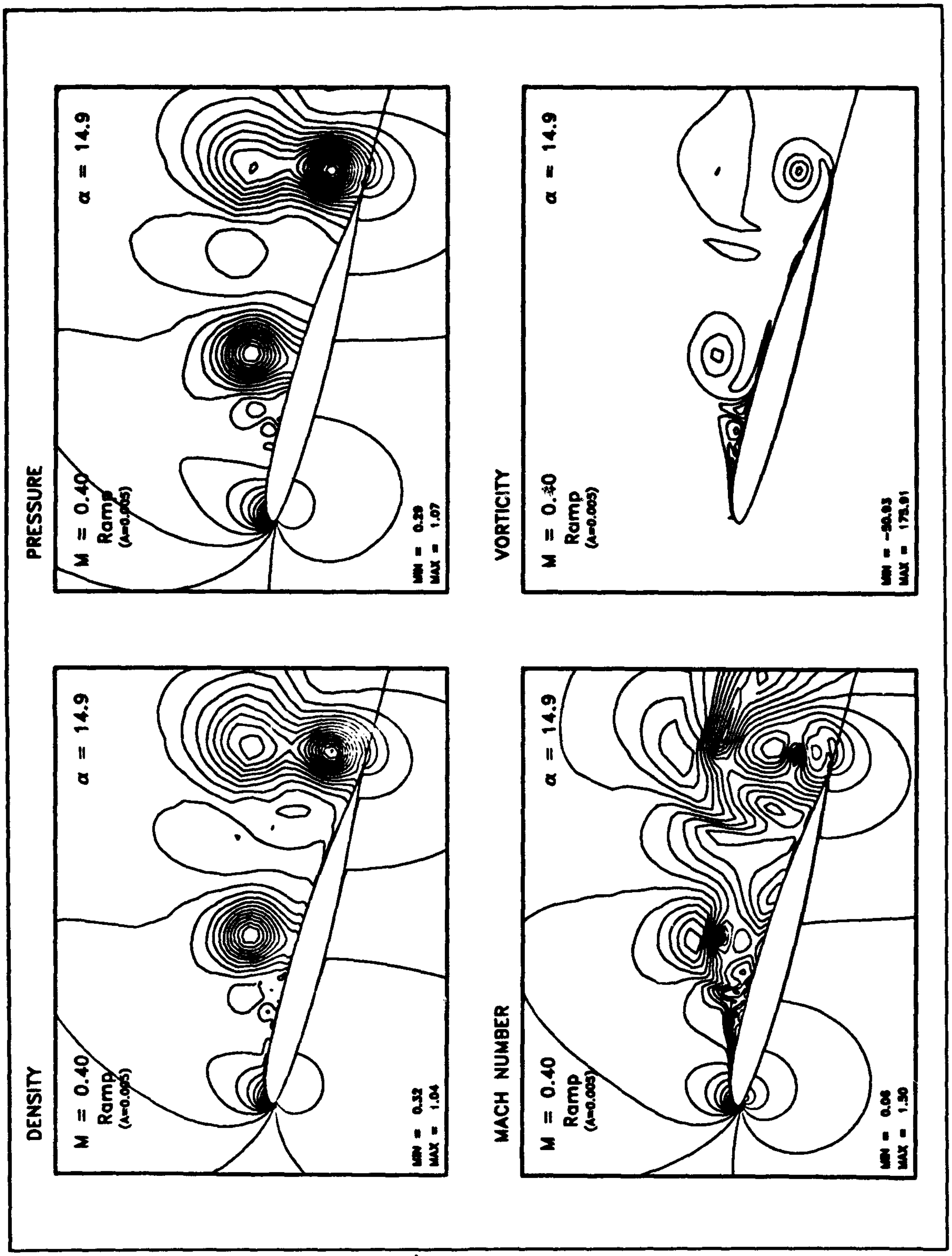


Figure 5.53
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

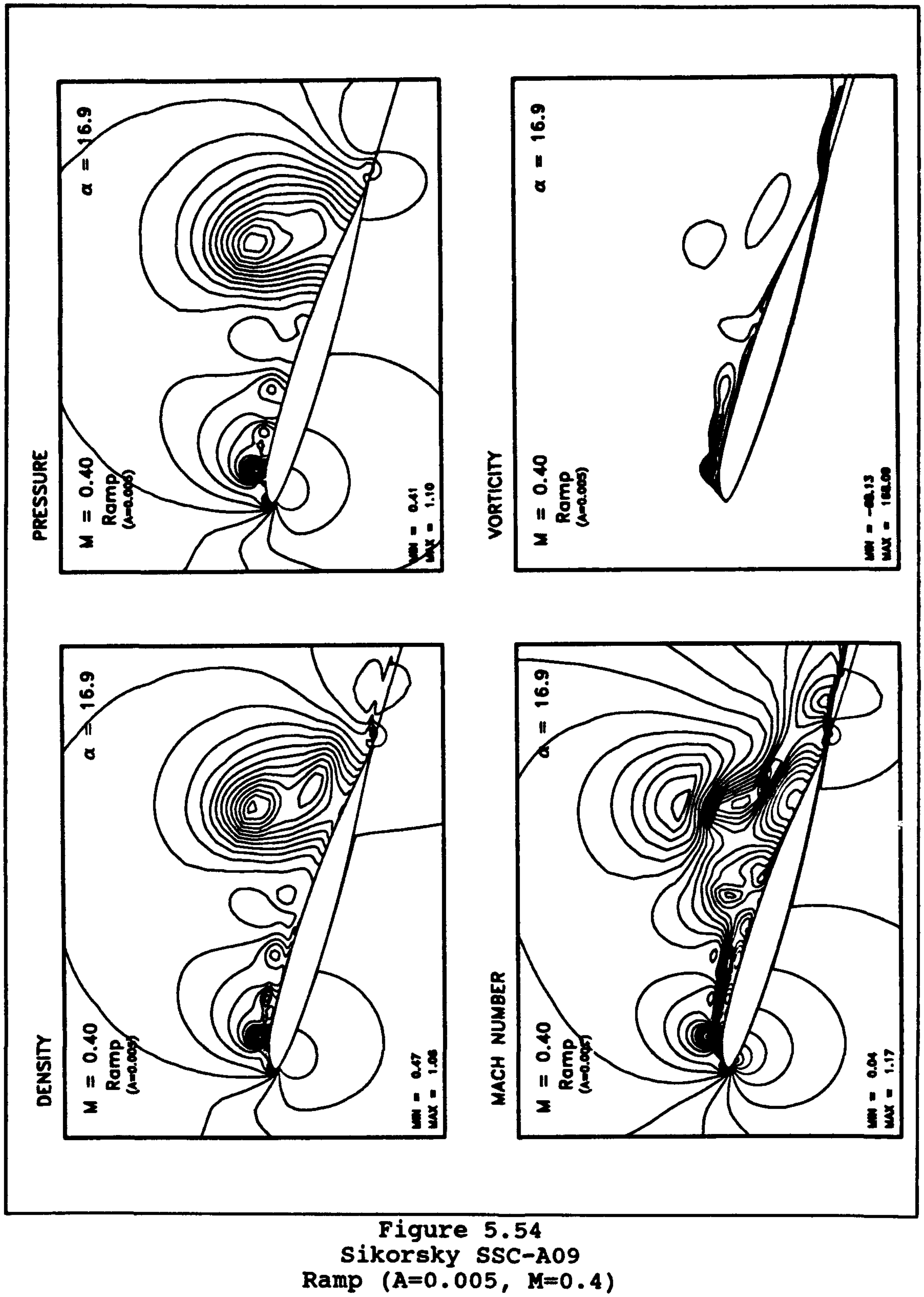


Figure 5.54
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

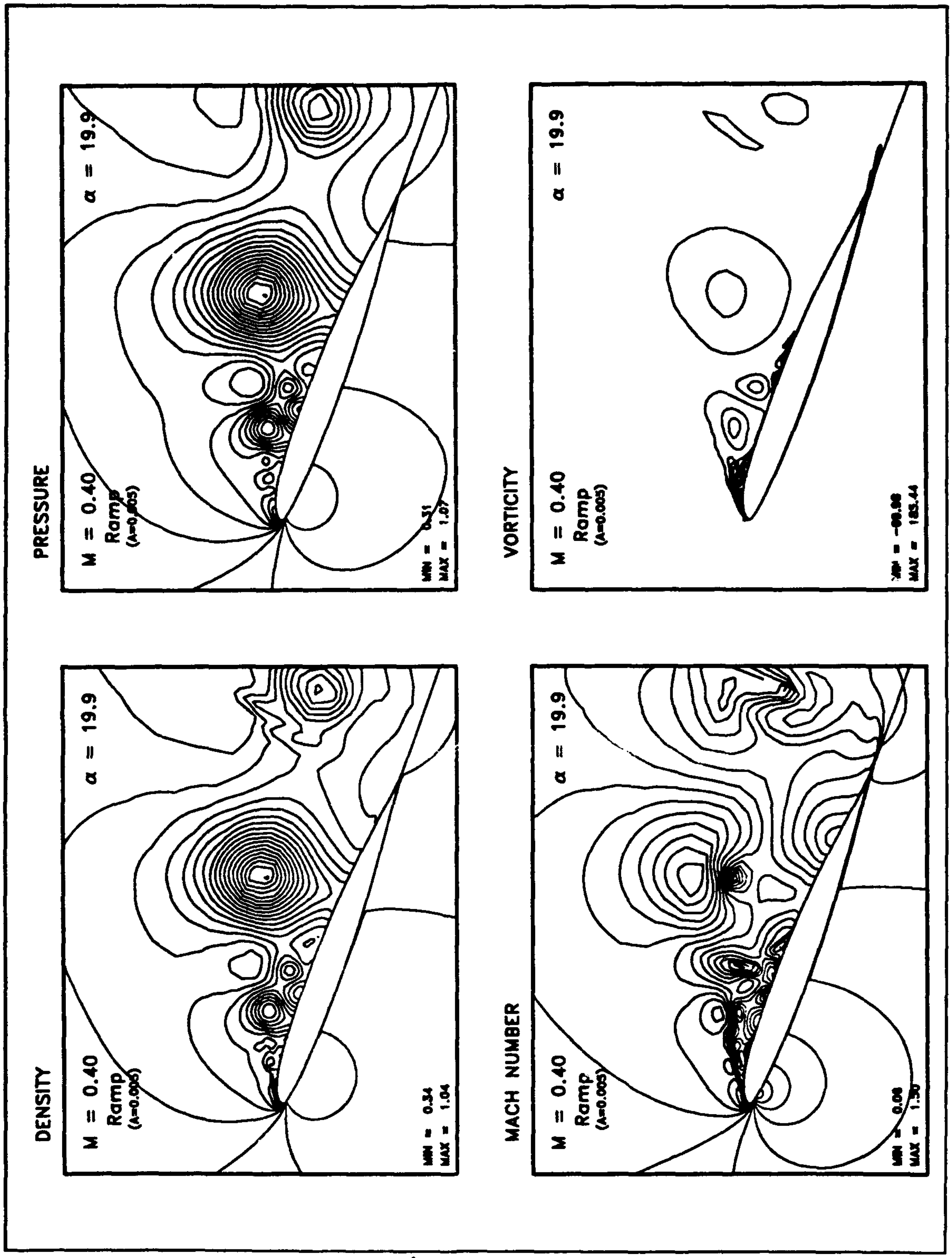


Figure 5.55
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

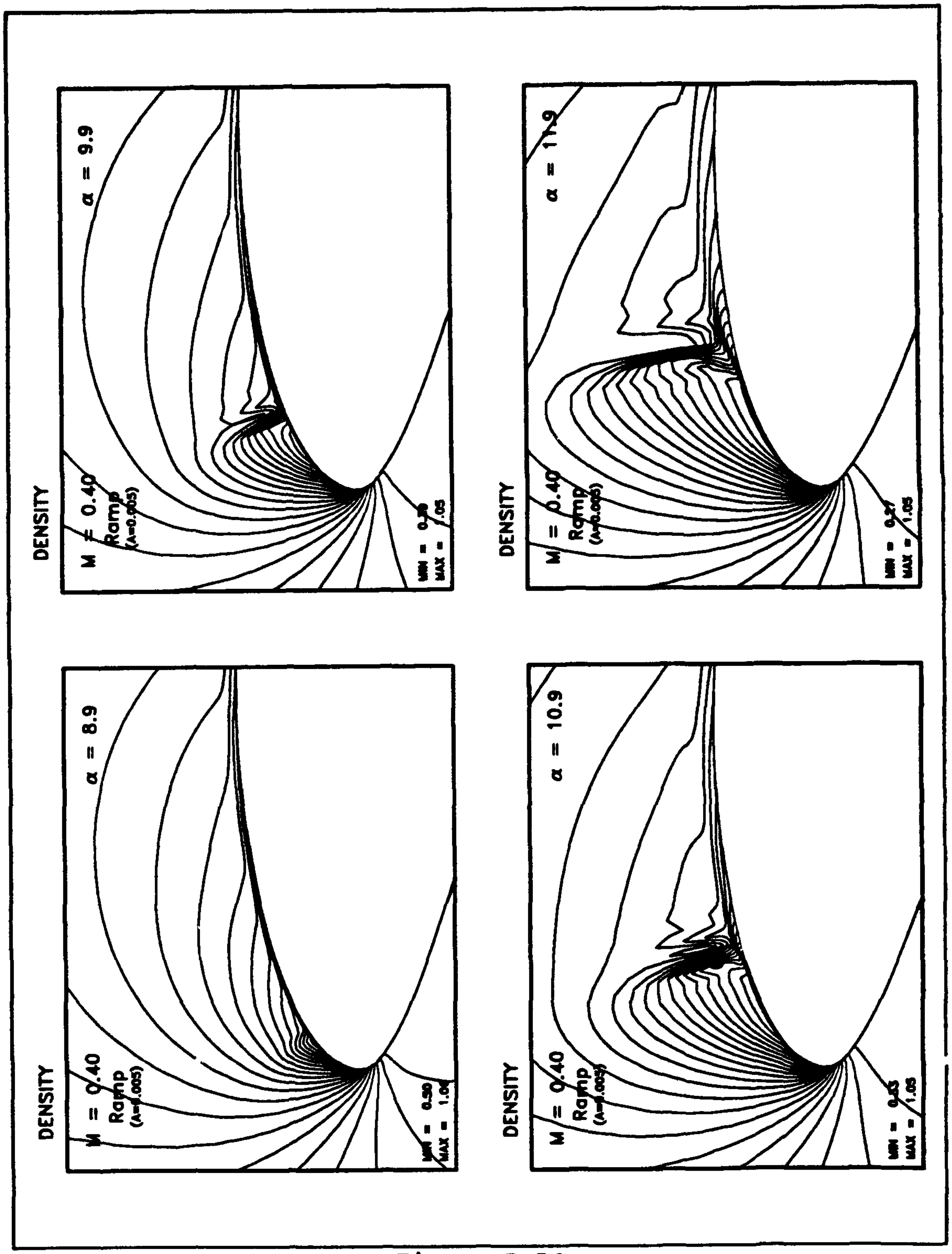


Figure 5.56
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

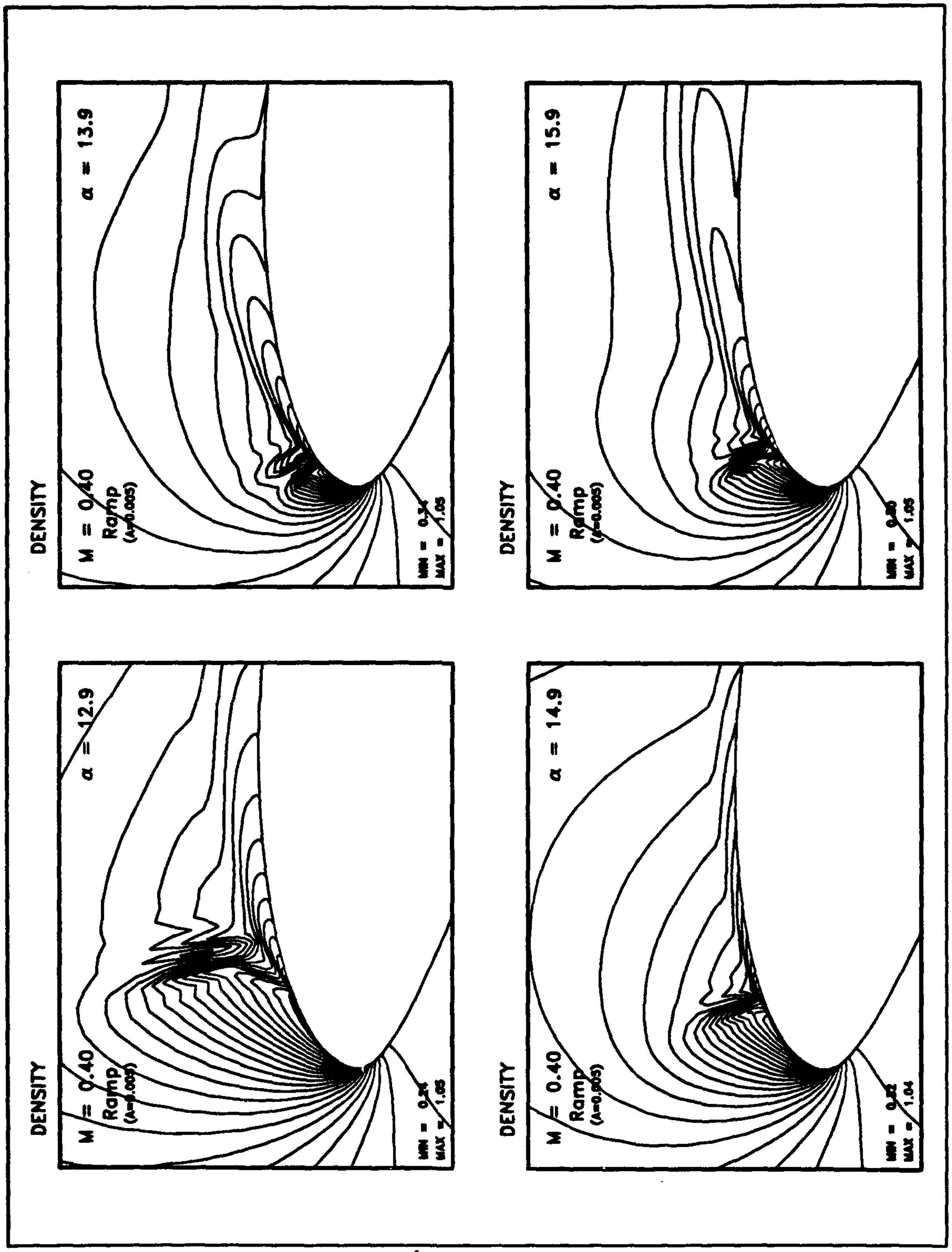


Figure 5.57
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

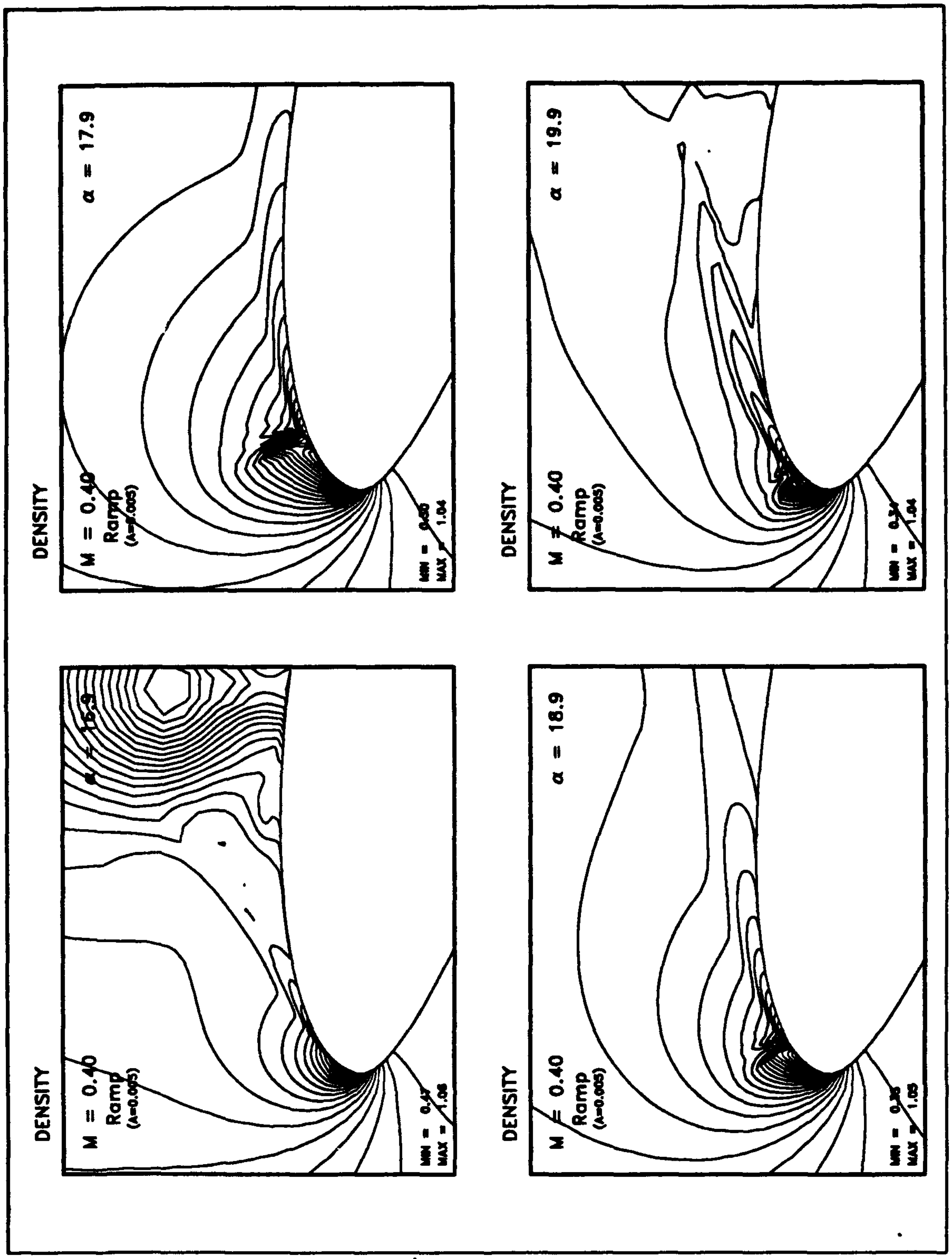


Figure 5.58
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

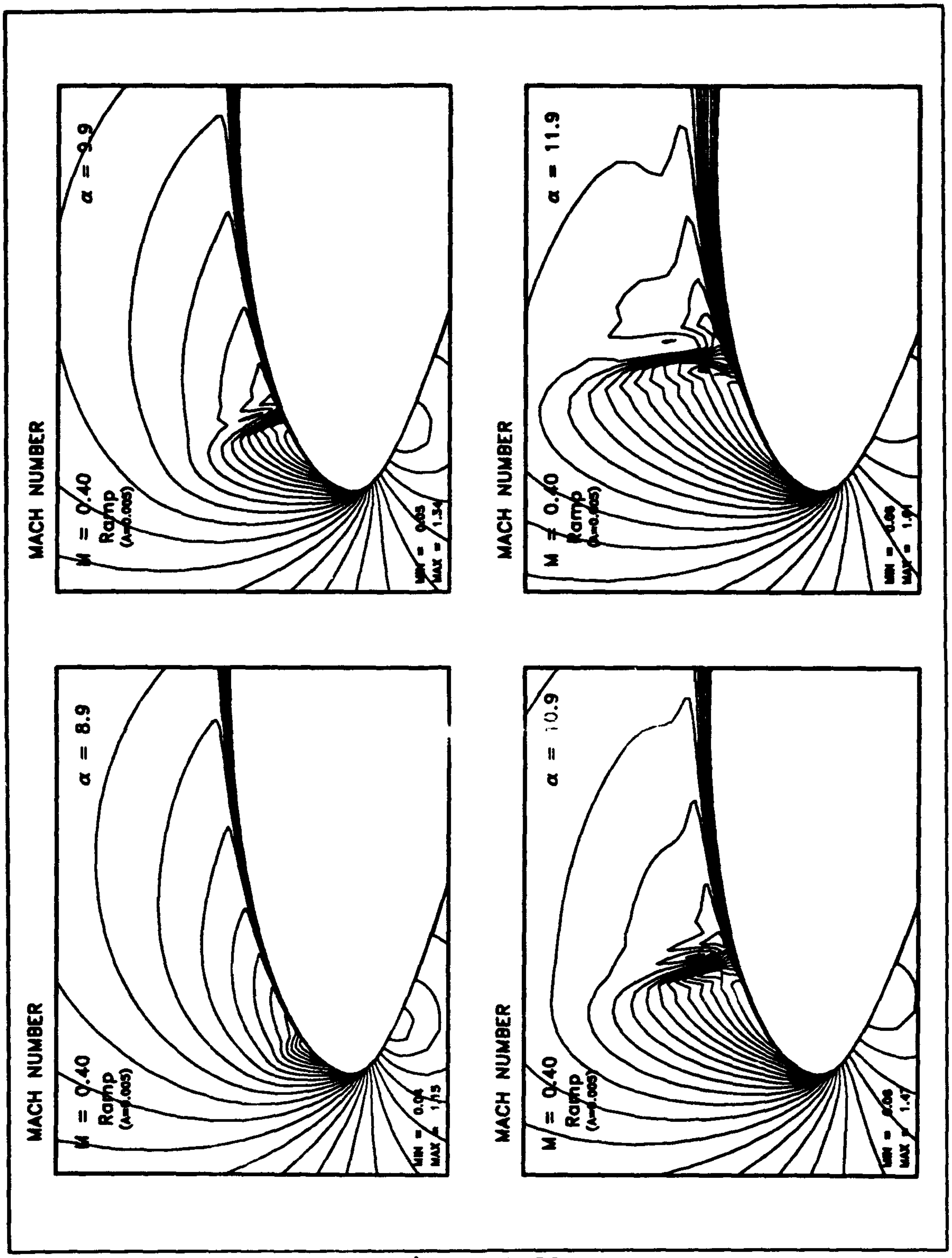


Figure 5.59
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

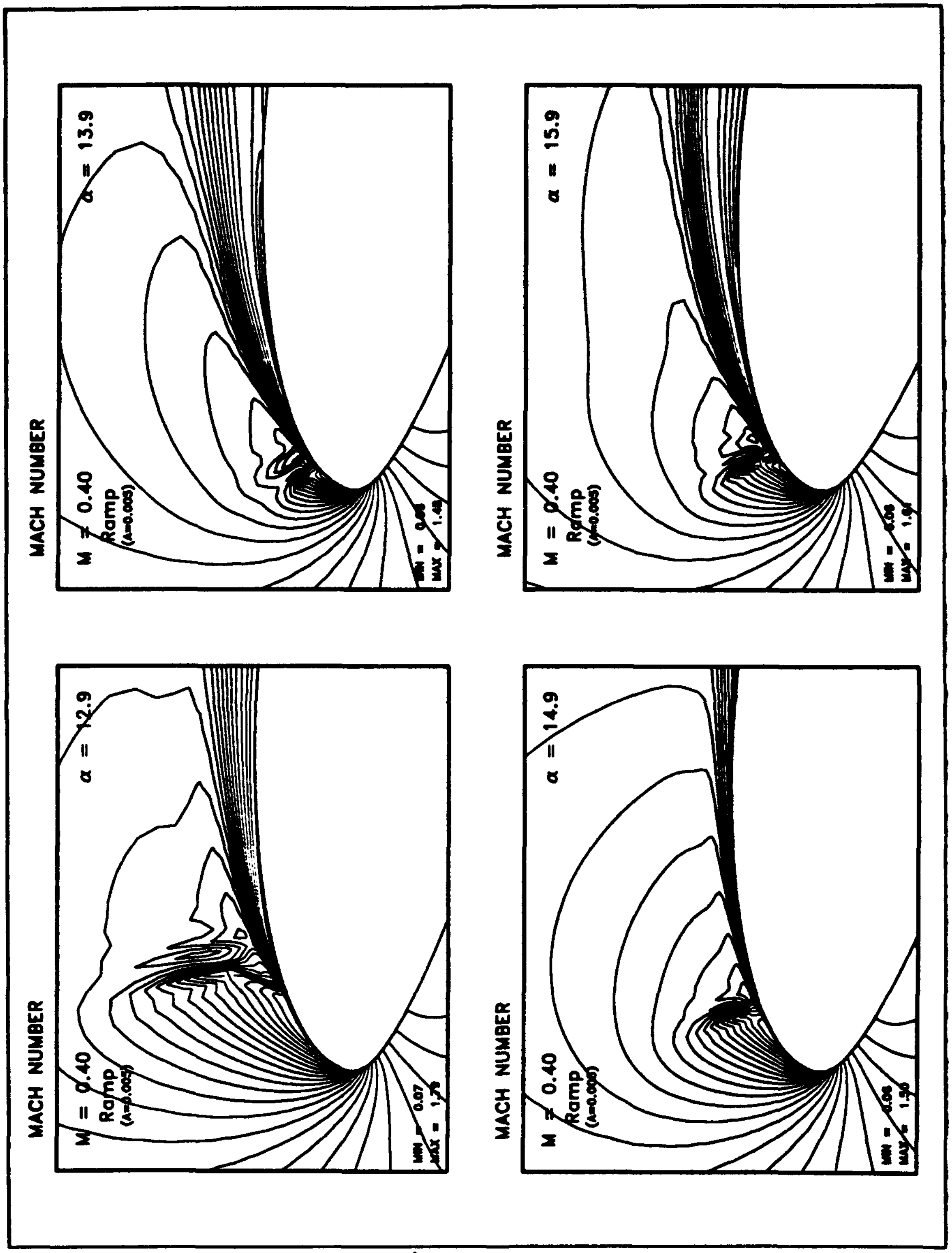


Figure 5.60
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

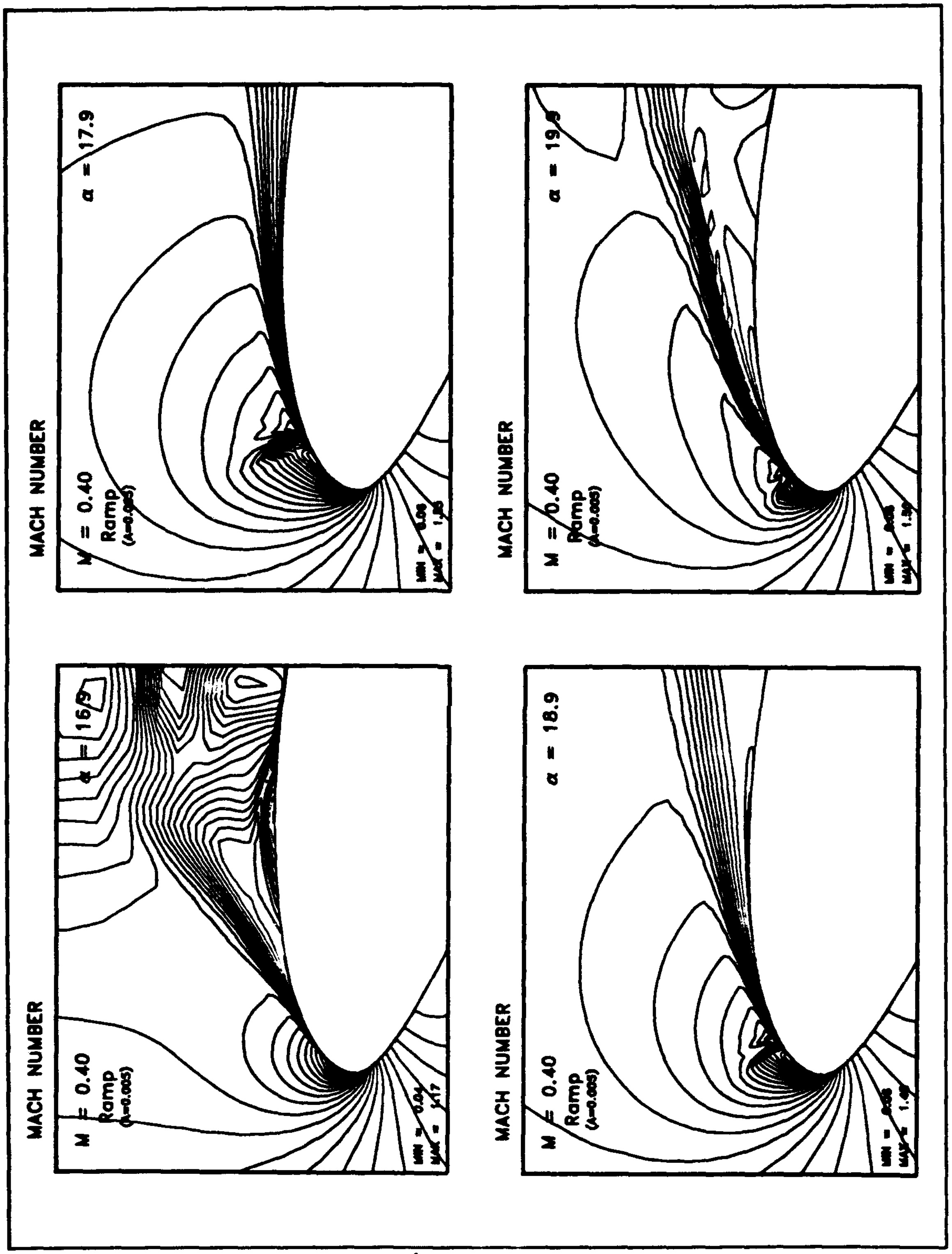


Figure 5.61
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

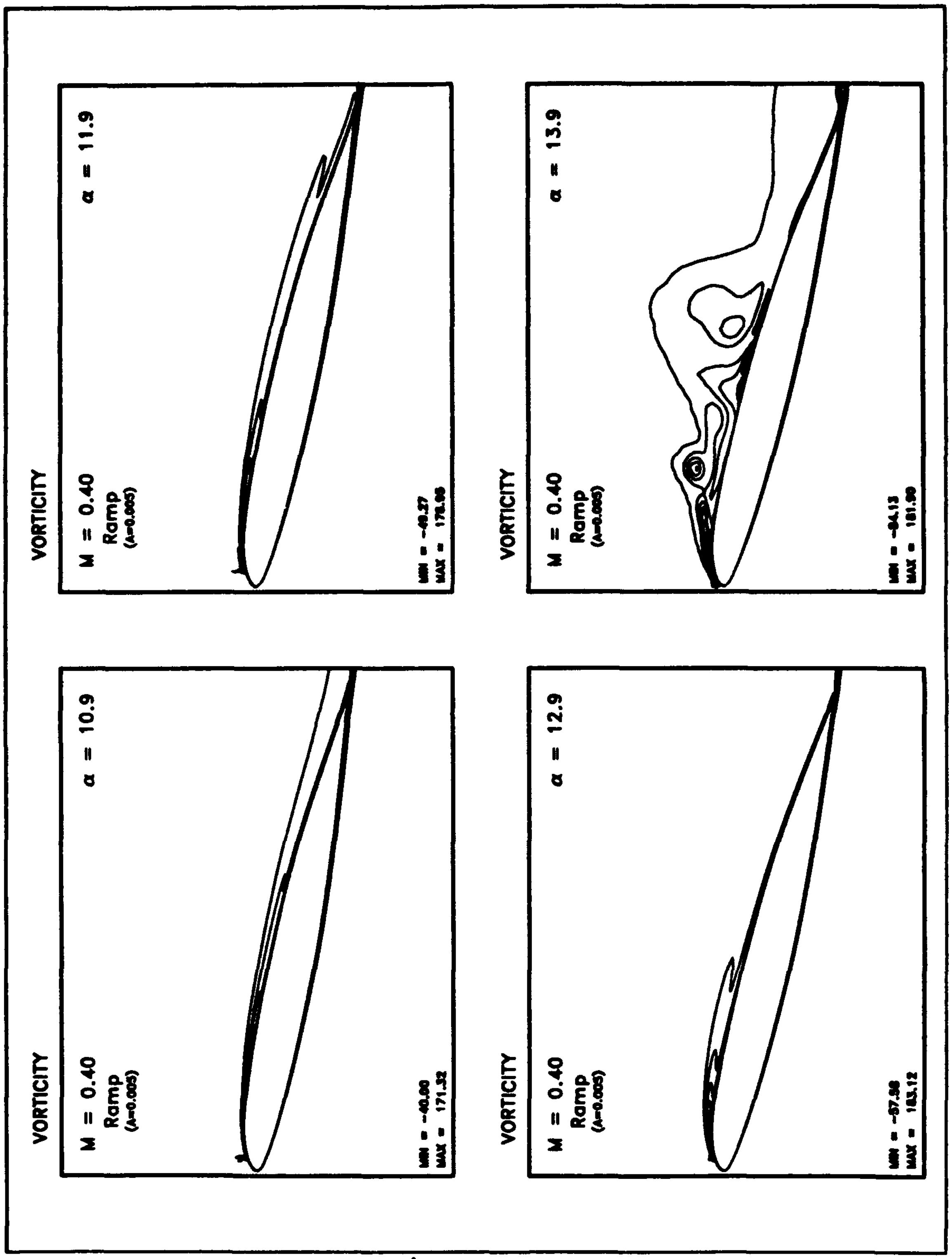


Figure 5.62
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

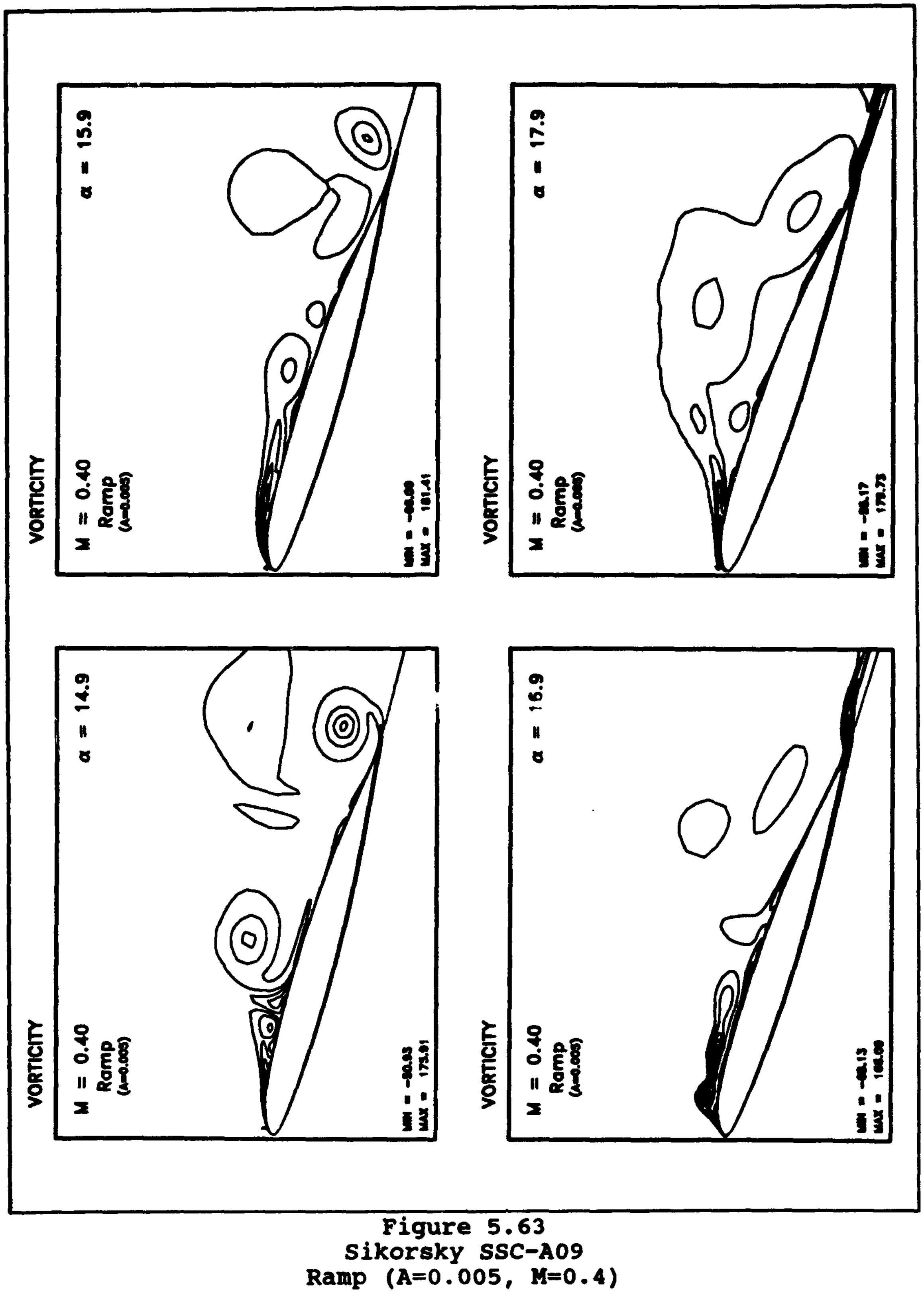


Figure 5.63
Sikorsky SSC-A09
Ramp (A=0.005, M=0.4)

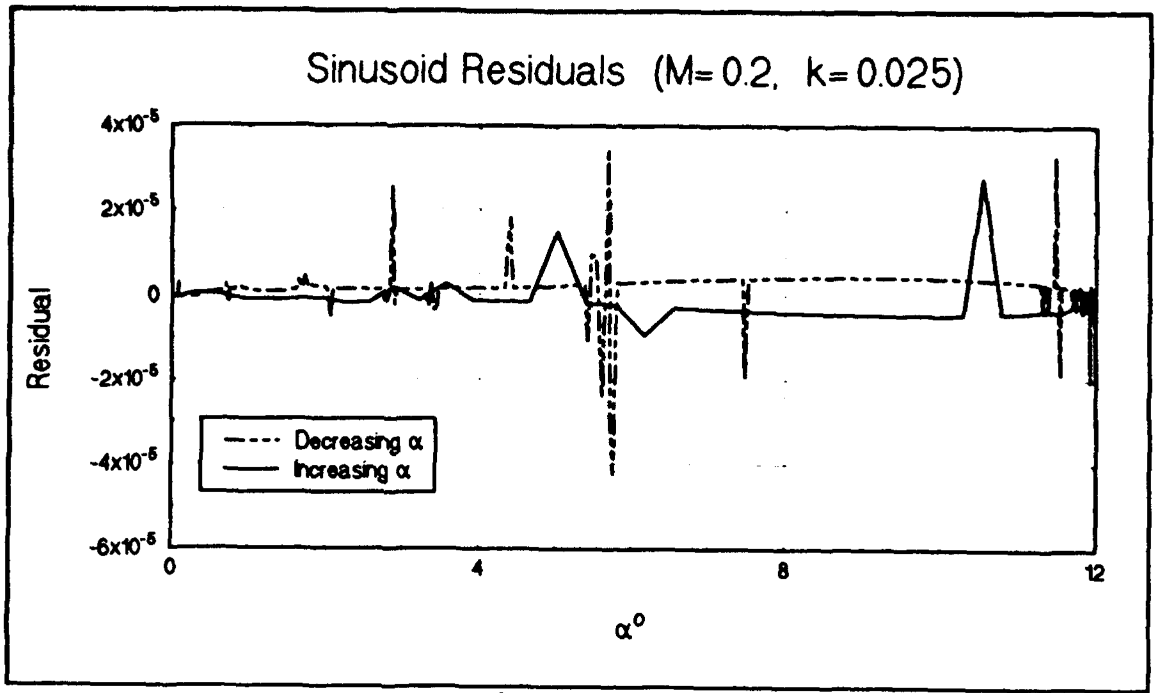


Figure 5.64
Sinusoid Residuals (k=0.025, M=0.2)

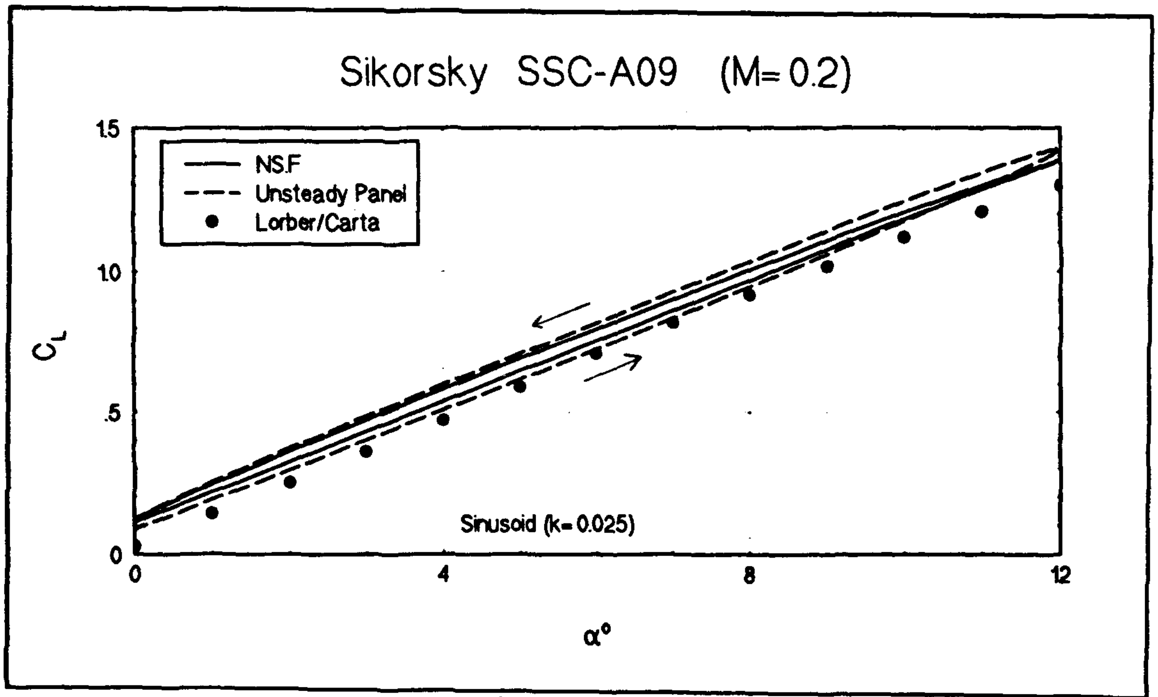


Figure 5.65
Sinusoid $C_{L\alpha}$ (k=0.025, M=0.2)

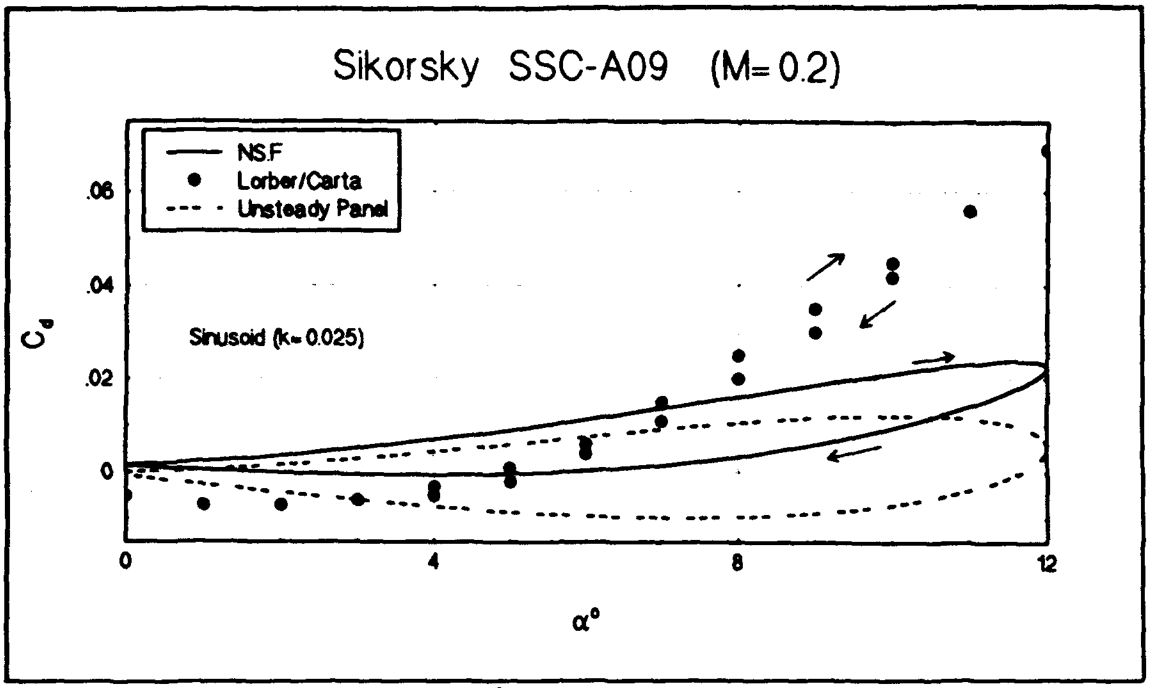


Figure 5.66
Sinusoid C_{ds} (k=0.025, M=0.2)

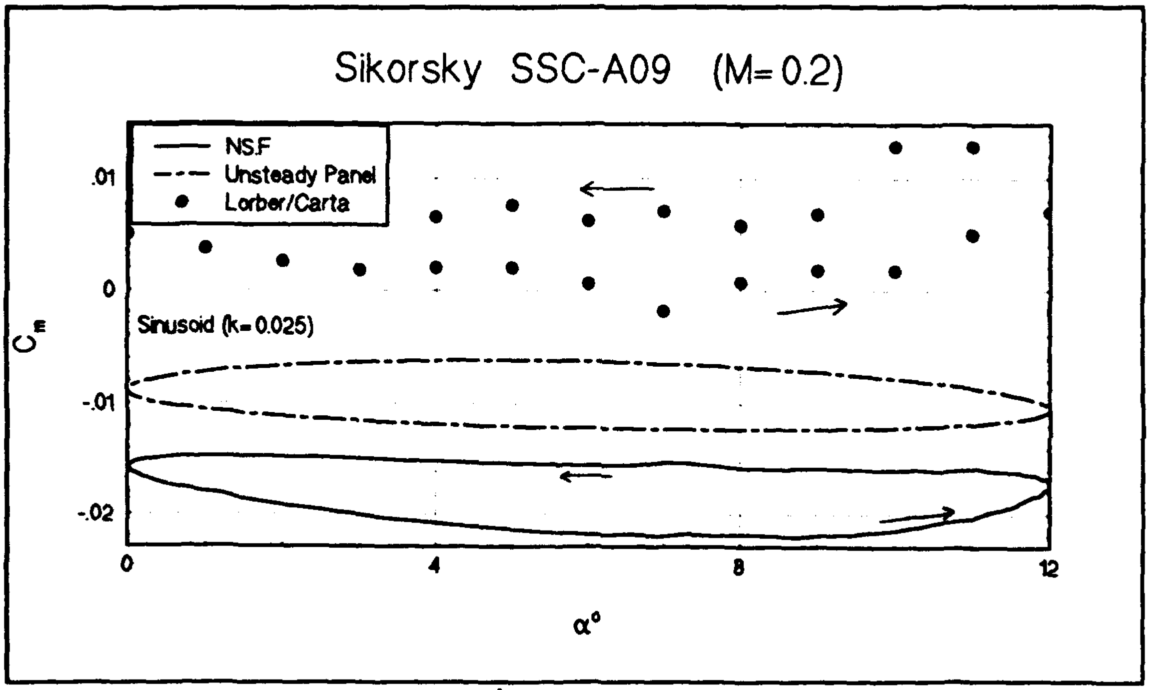


Figure 5.67
Sinusoid C_{Ms} (k=0.025, M=0.2)

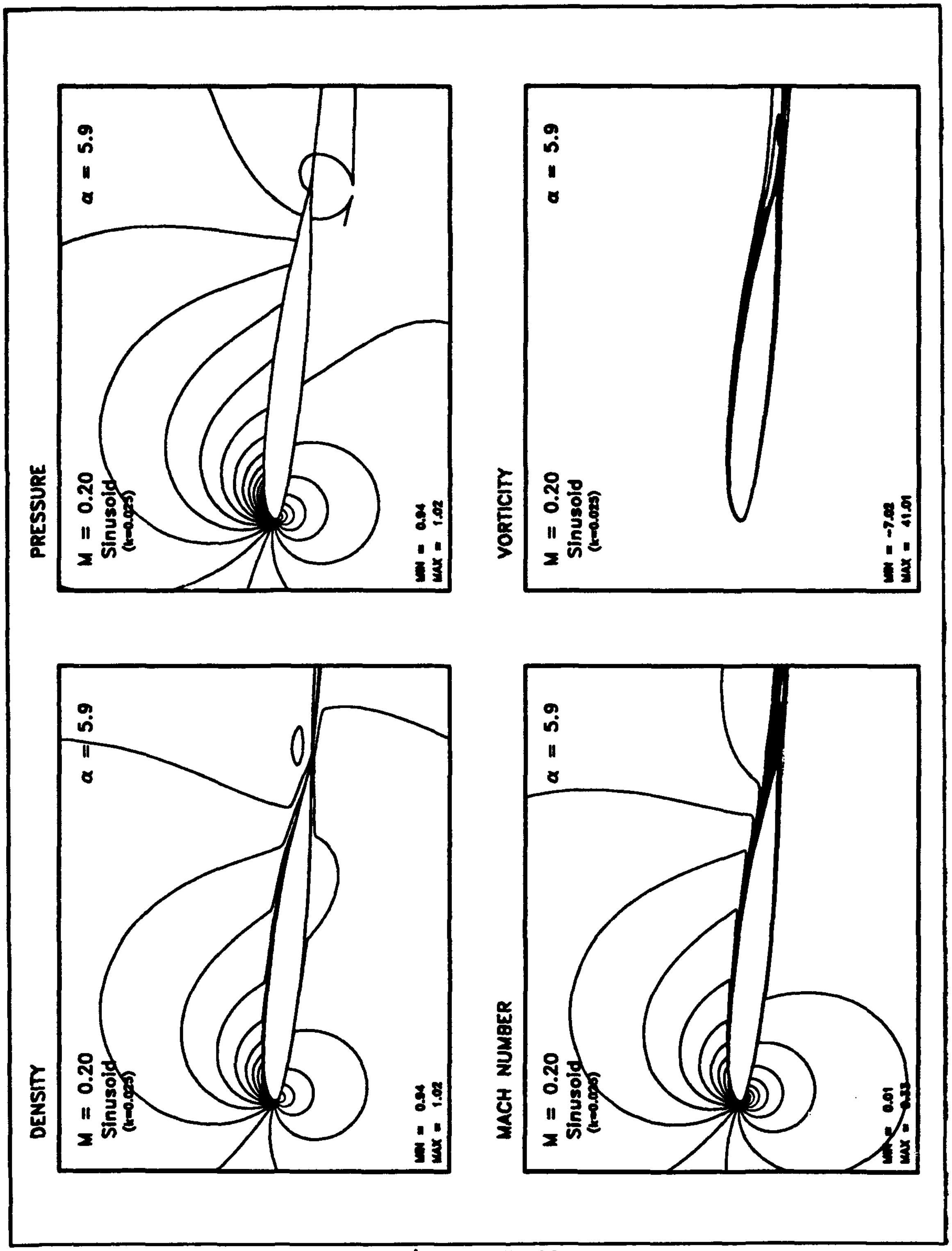


Figure 5.68
Sikorsky SSC-A09
Sinusoid (UP) (k=0.025, M=0.2)

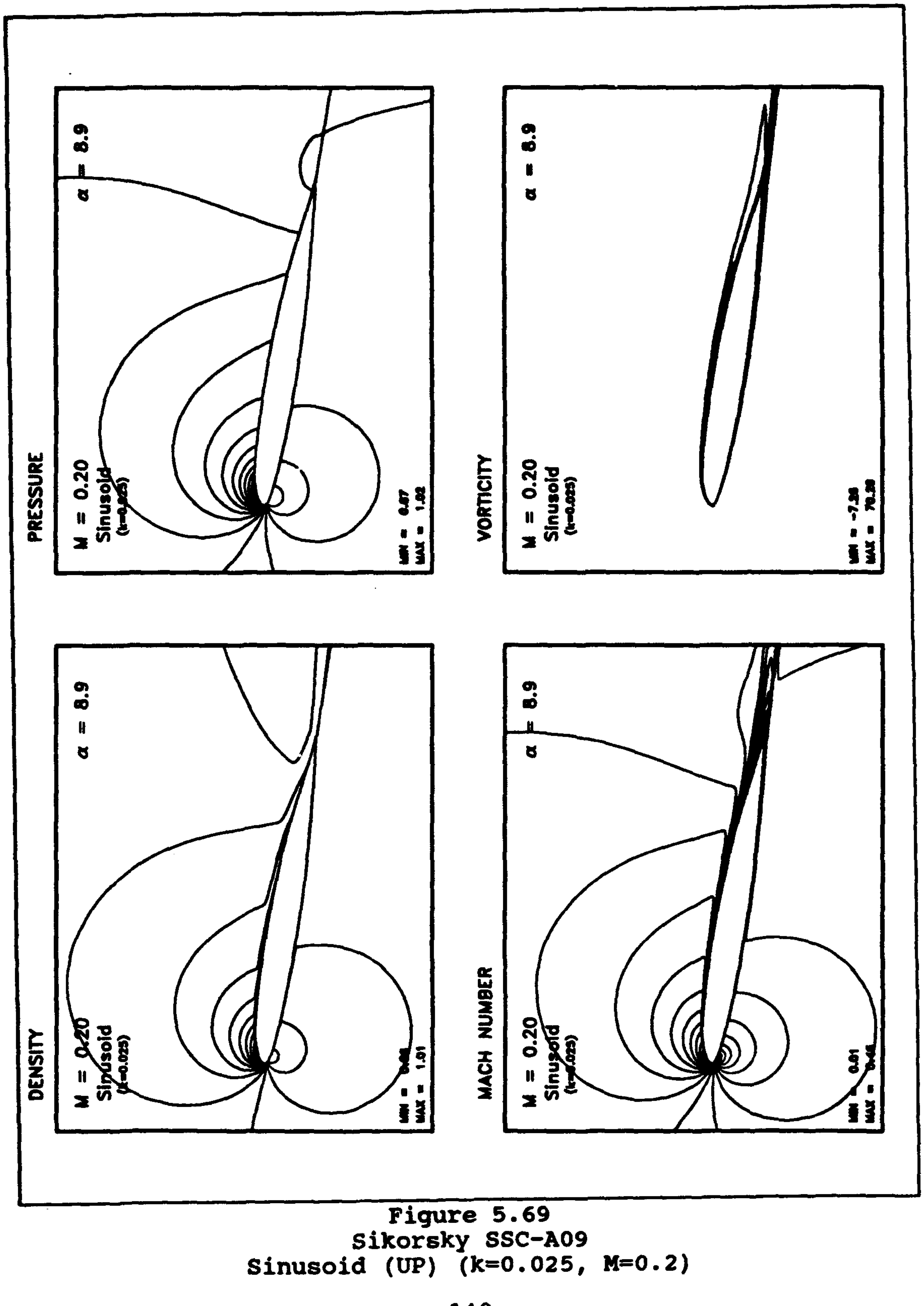


Figure 5.69
Sikorsky SSC-A09
Sinusoid (UP) (k=0.025, M=0.2)

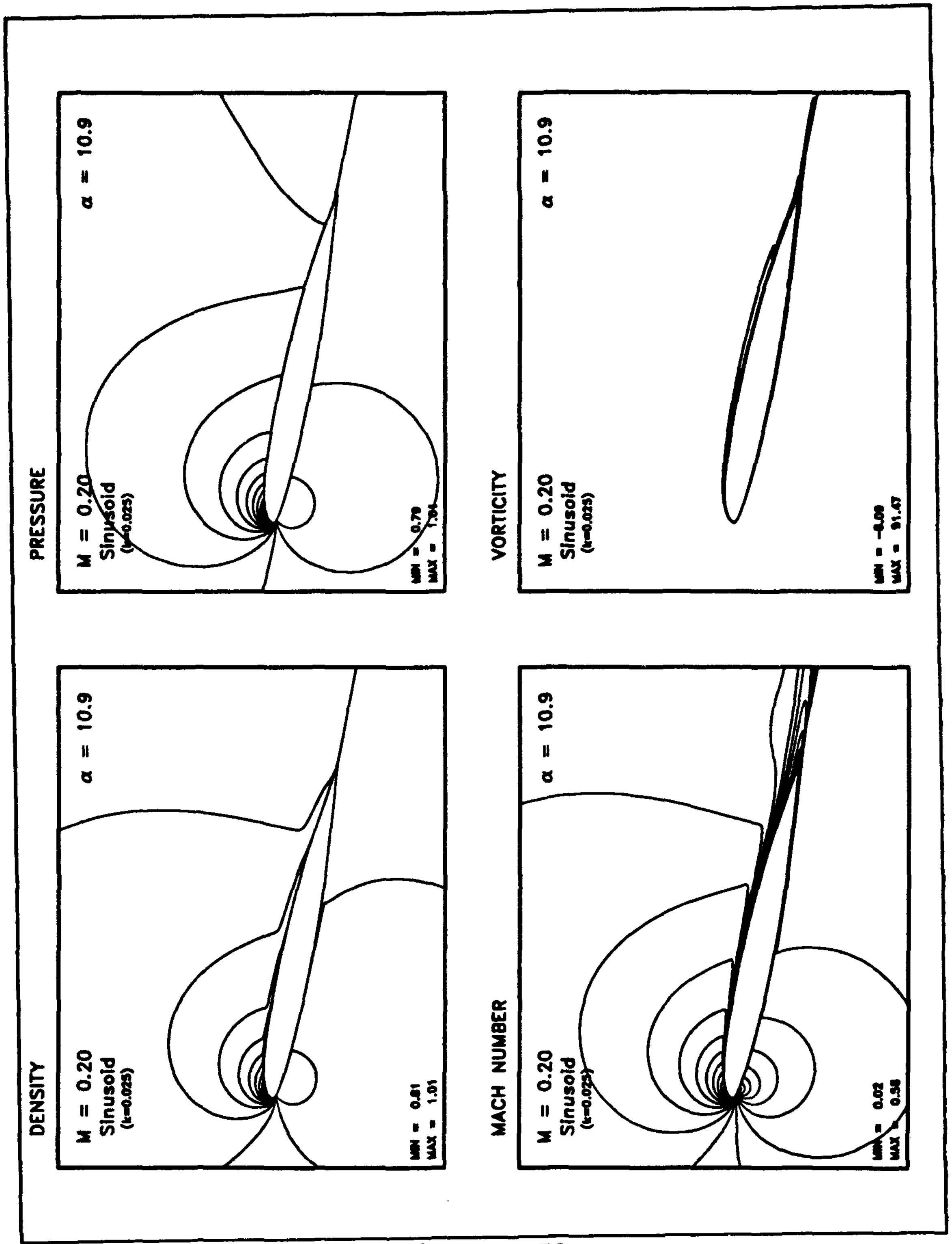


Figure 5.70
Sikorsky SSC-A09
Sinusoid (UP) (k=0.025, M=0.2)

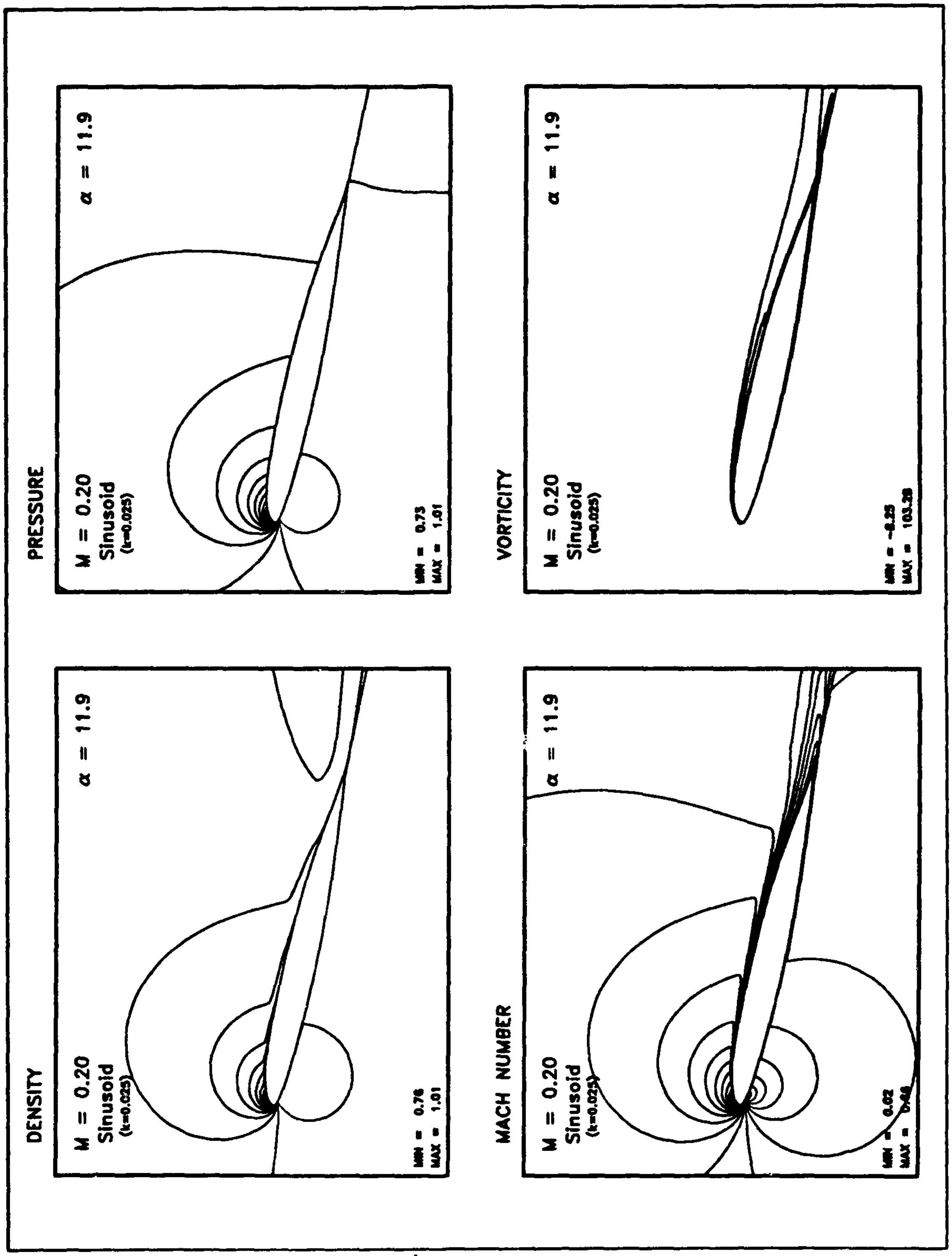


Figure 5.71
Sikorsky SSC-A09
Sinusoid (UP) (k=0.025, M=0.2)

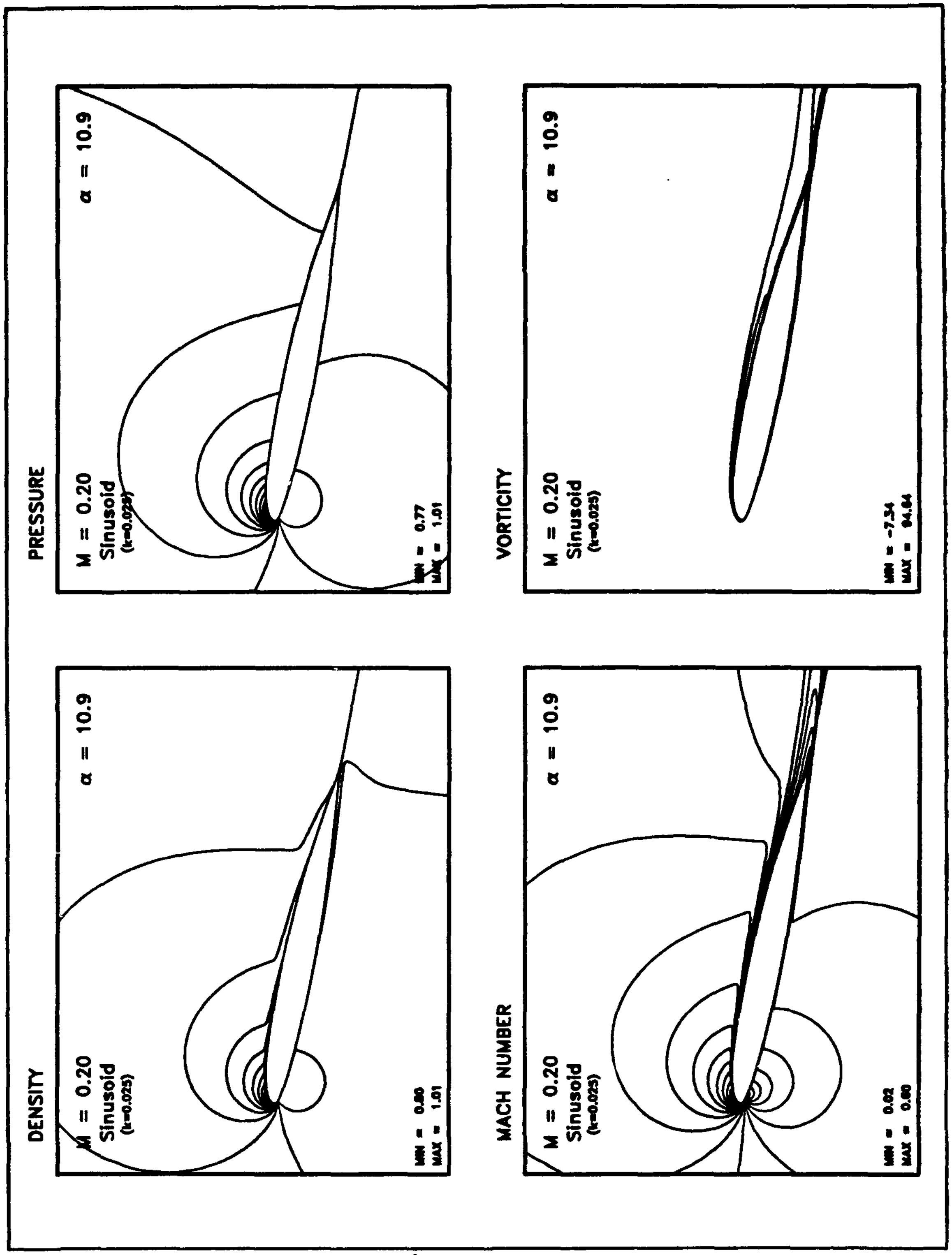


Figure 5.72
Sikorsky SSC-A09
Sinusoid (DOWN) (k=0.025, M=0.2)

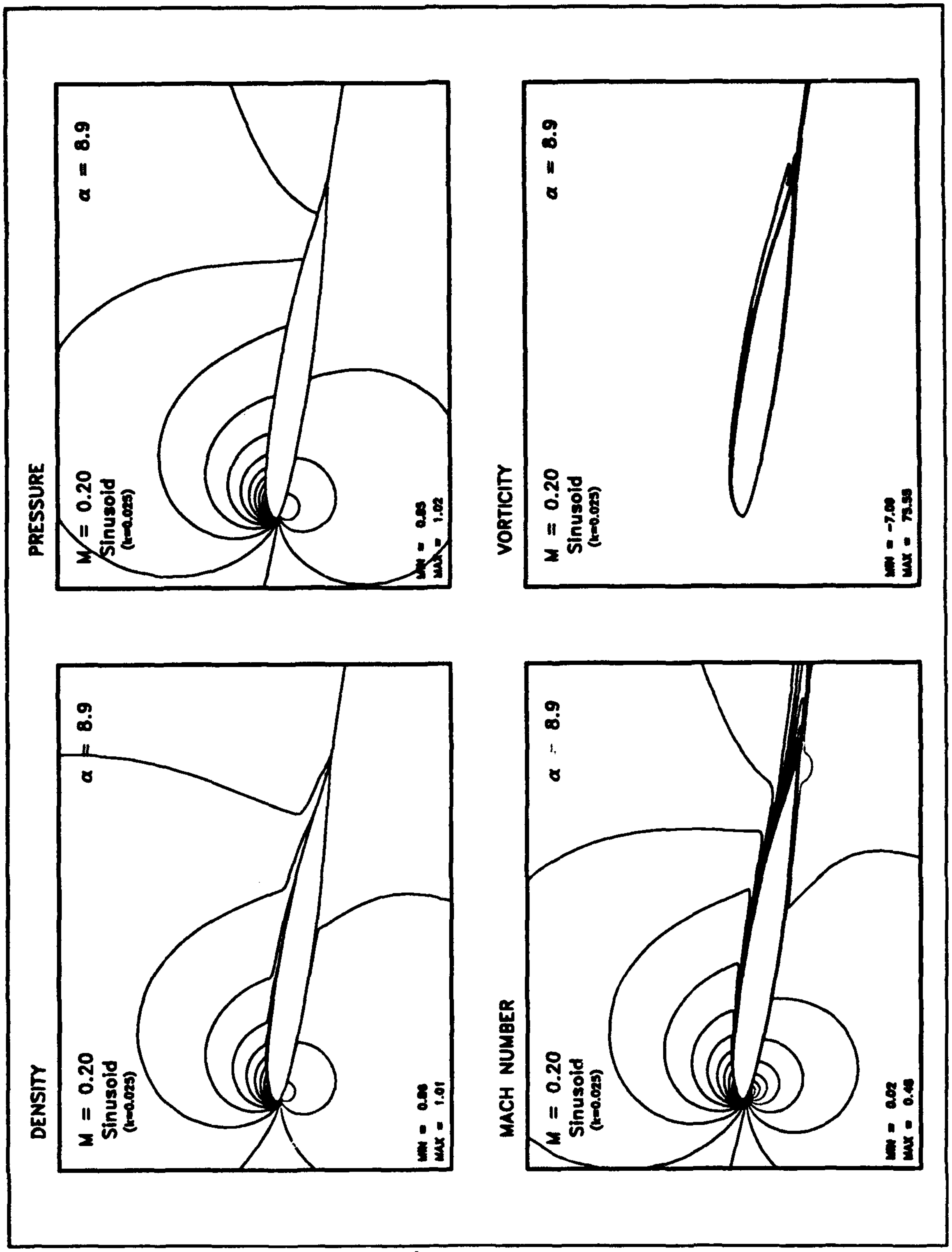


Figure 5.73
Sikorsky SSC-A09
Sinusoid (DOWN) (k=0.025, M=0.2)

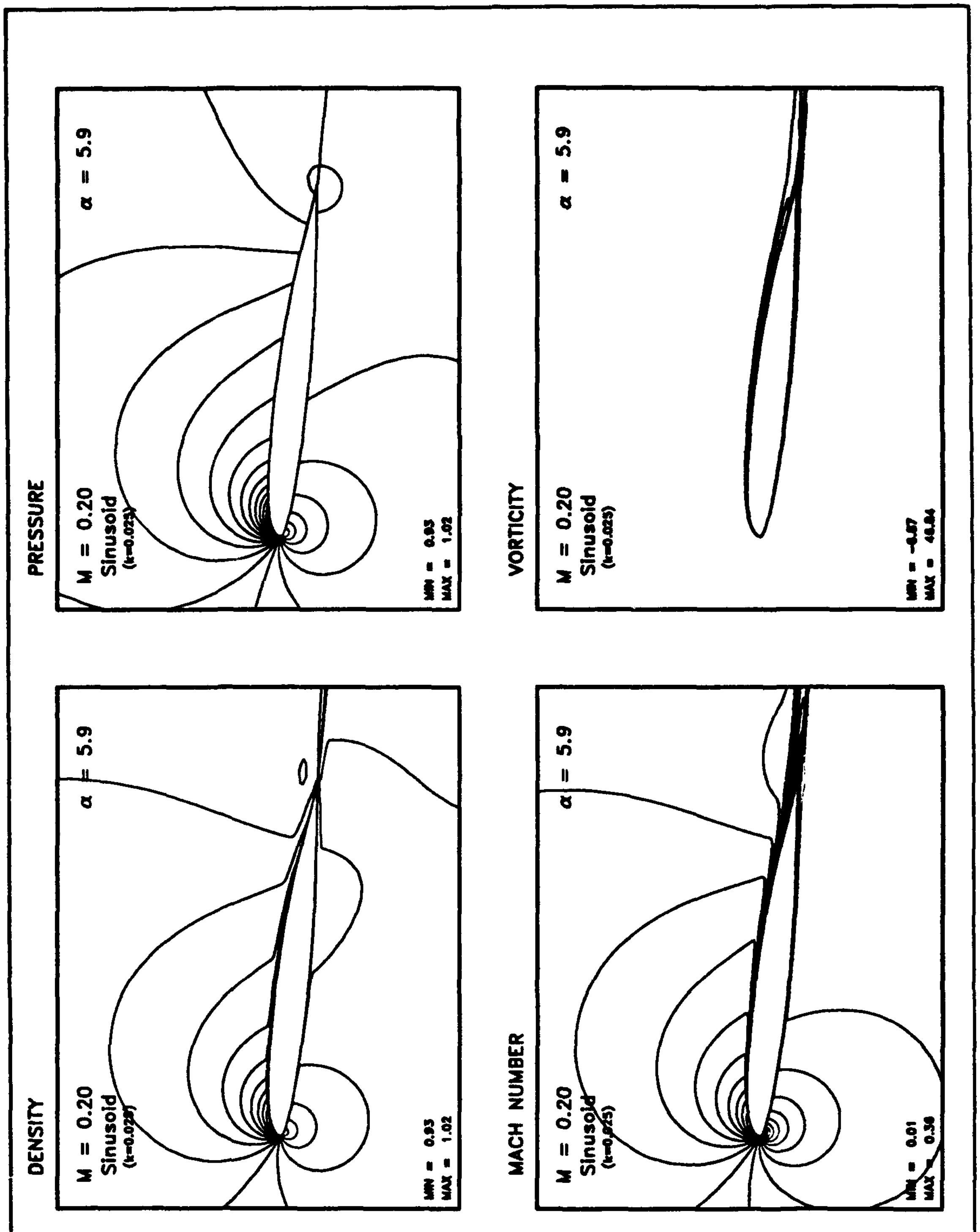


Figure 5.74
Sikorsky SSC-A09
Sinusoid (DOWN) (k=0.025, M=0.2)

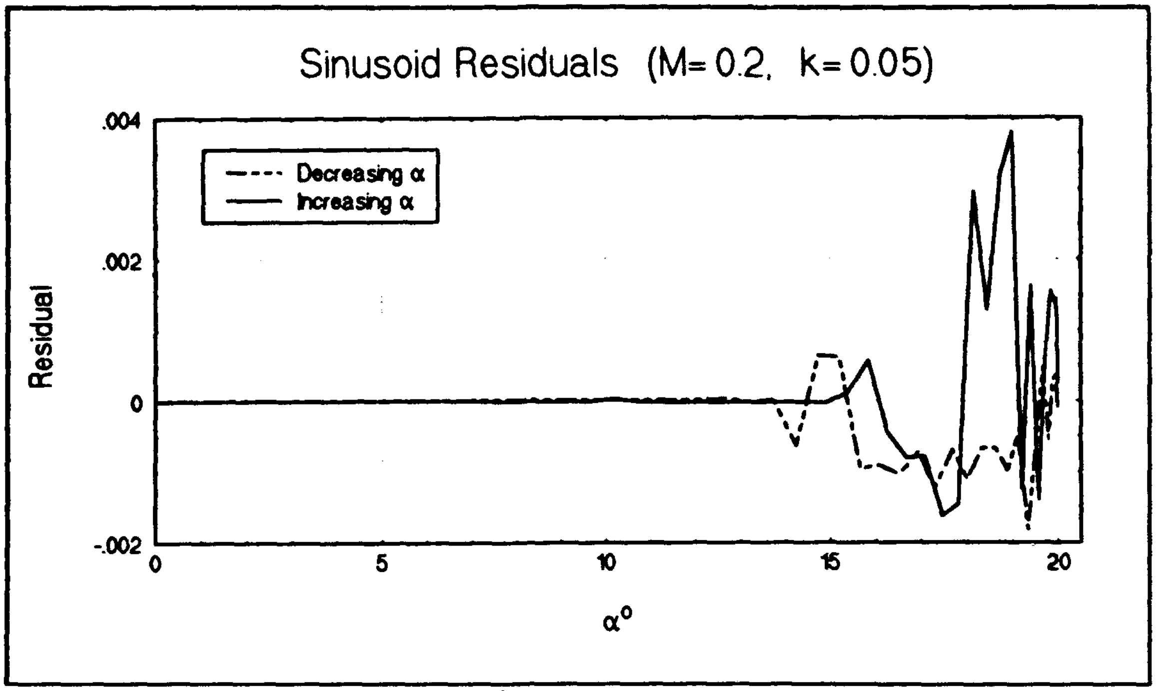


Figure 5.75
Sinusoid Residuals (k=0.05, M=0.2)

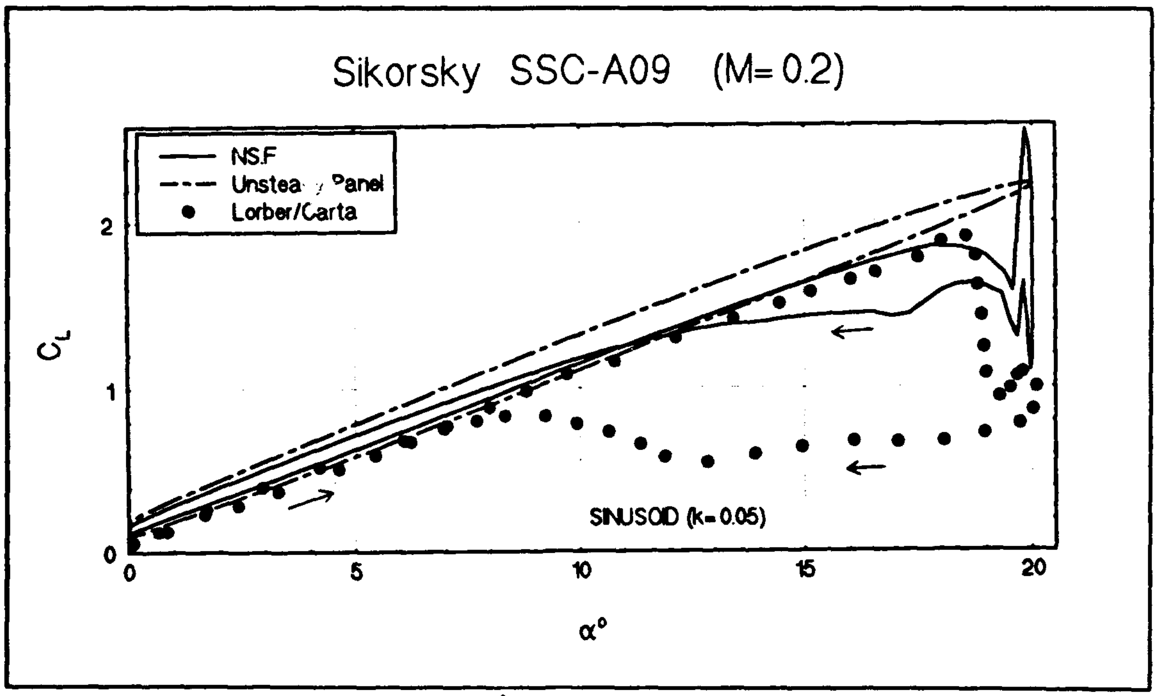


Figure 5.76
Sinusoid $C_{L\alpha}$ (k=0.05, M=0.2)

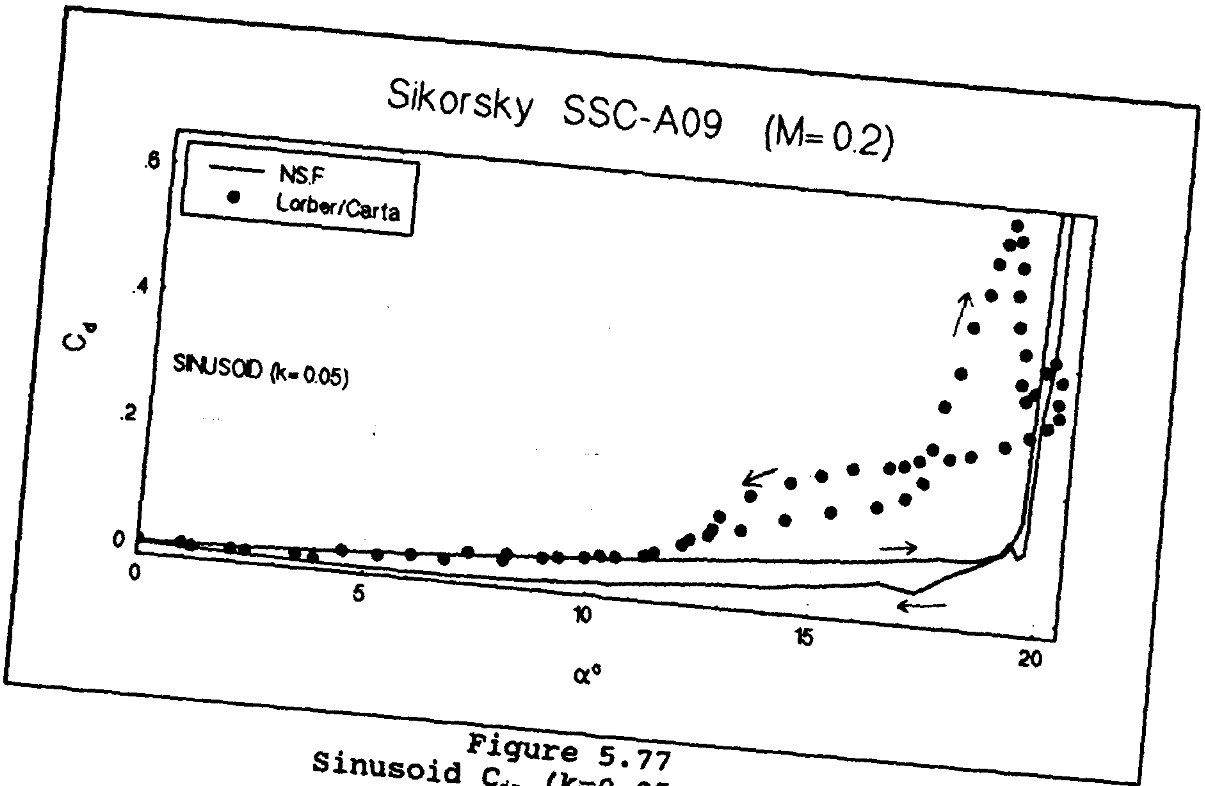


Figure 5.77
Sinusoid C_{ds} (k=0.05, M=0.2)

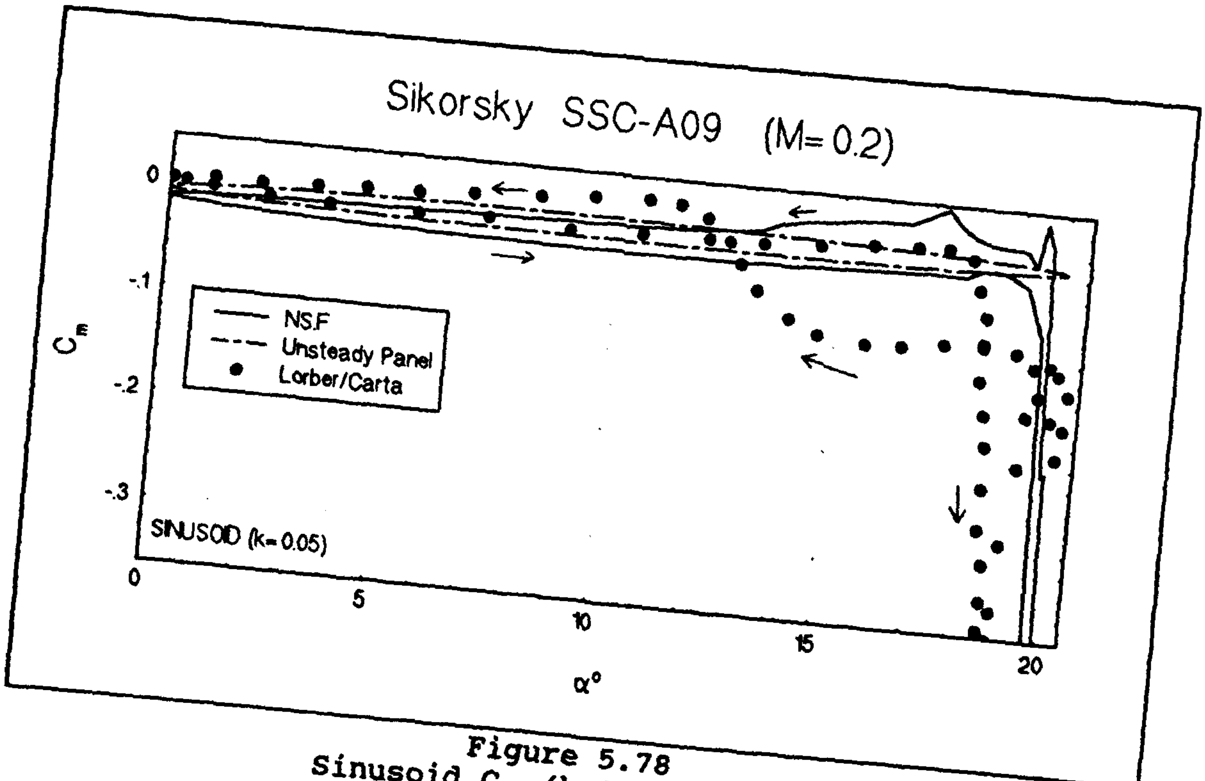


Figure 5.78
Sinusoid C_{Ms} (k=0.05, M=0.2)

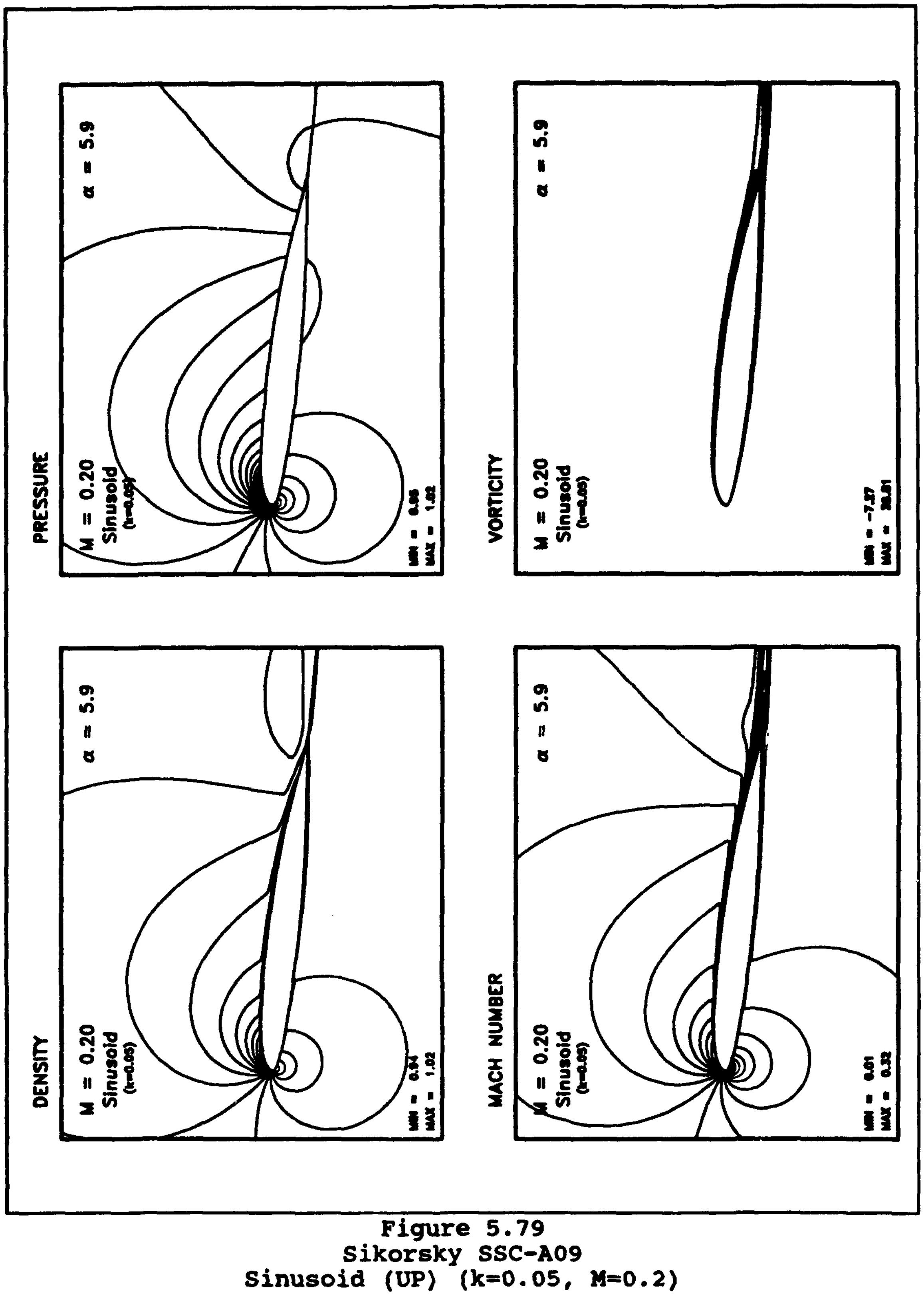


Figure 5.79
Sikorsky SSC-A09
Sinusoid (UP) (k=0.05, M=0.2)

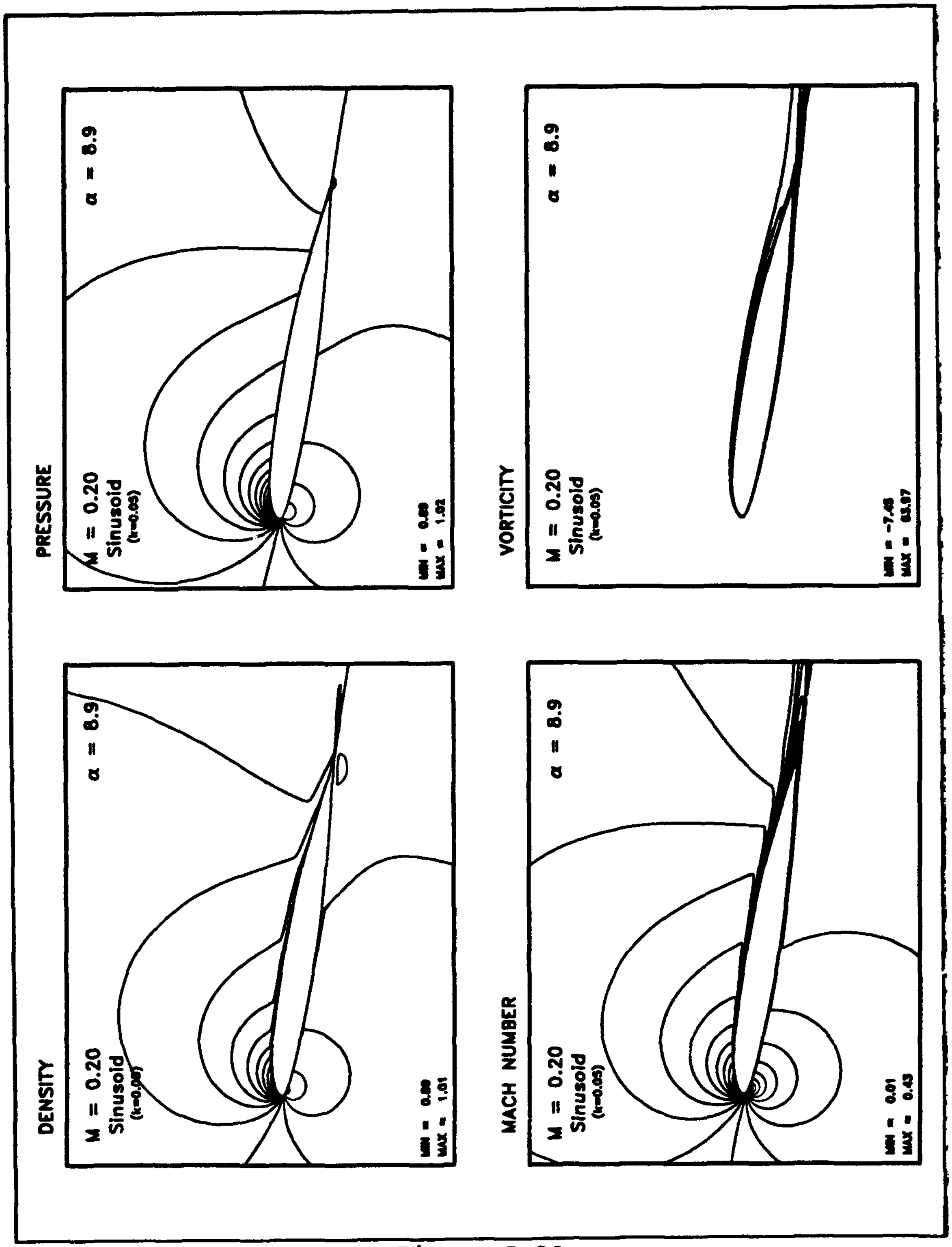


Figure 5.80
Sikorsky SSC-A09
Sinusoid (UP) (k=0.05, M=0.2)

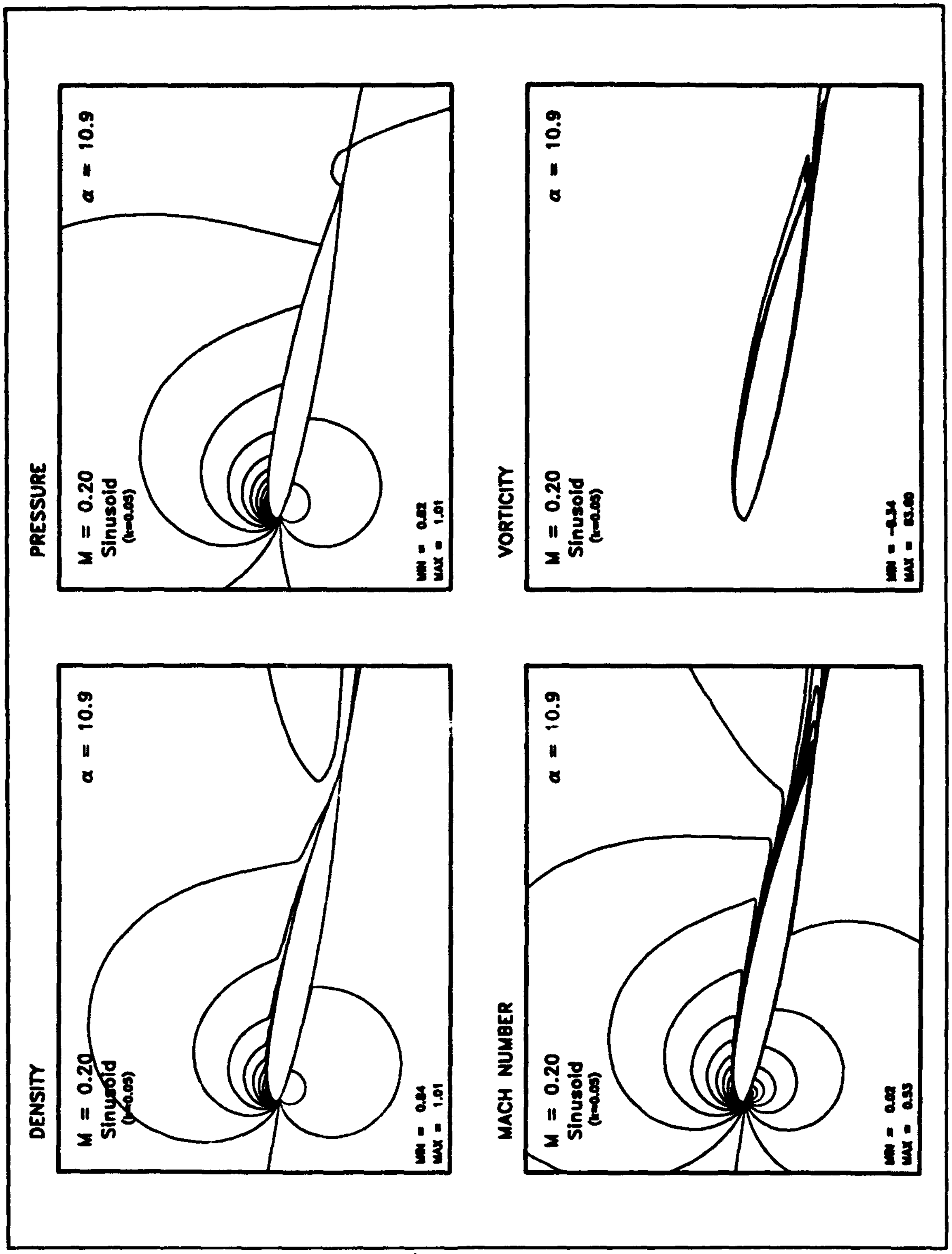


Figure 5.81
Sikorsky SSC-A09
Sinusoid (UP) (k=0.05, M=0.2)

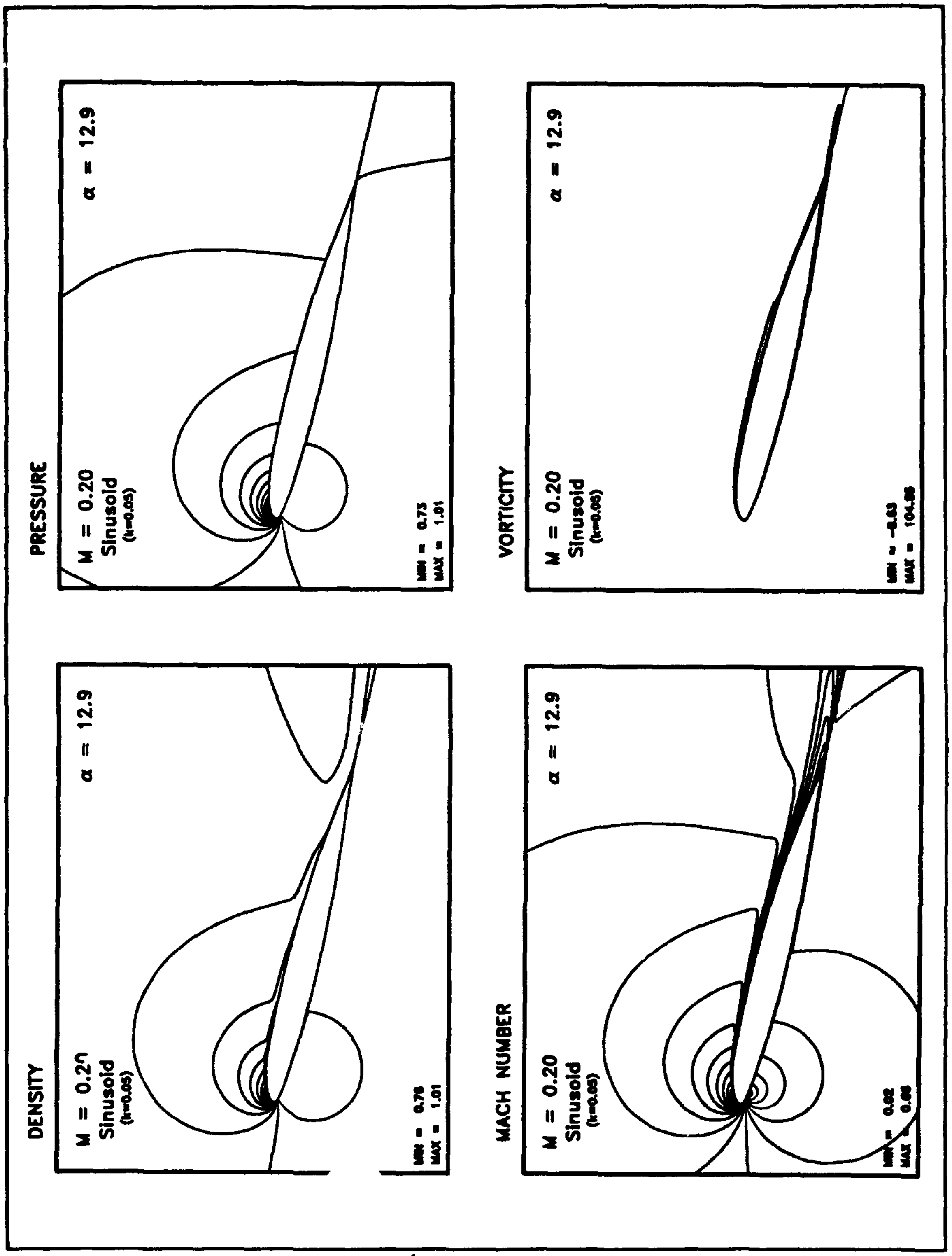


Figure 5.82
Sikorsky SSC-A09
Sinusoid (UP) (k=0.05, M=0.2)

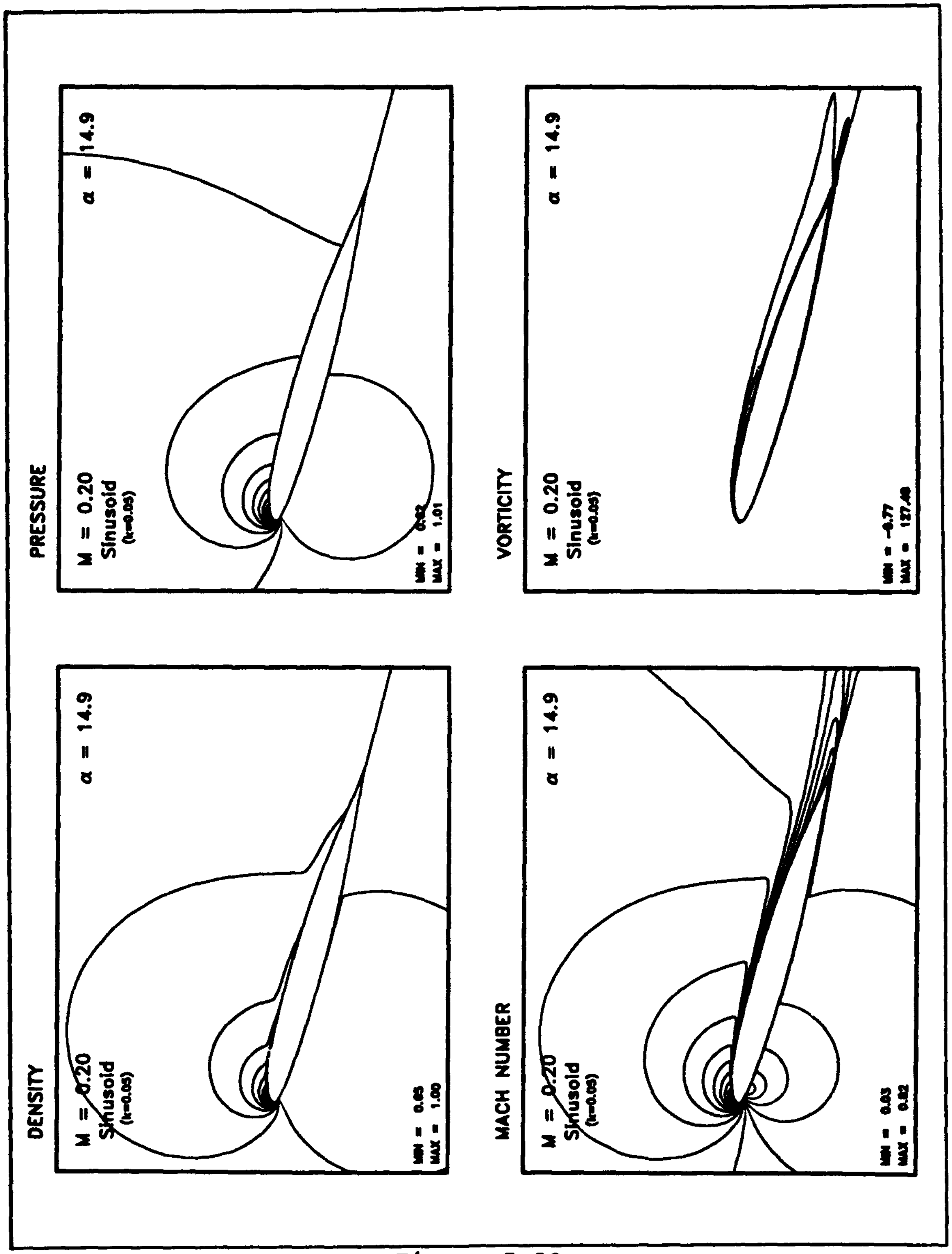


Figure 5.83
Sikorsky SSC-A09
Sinusoid (UP) (k=0.05, M=0.2)

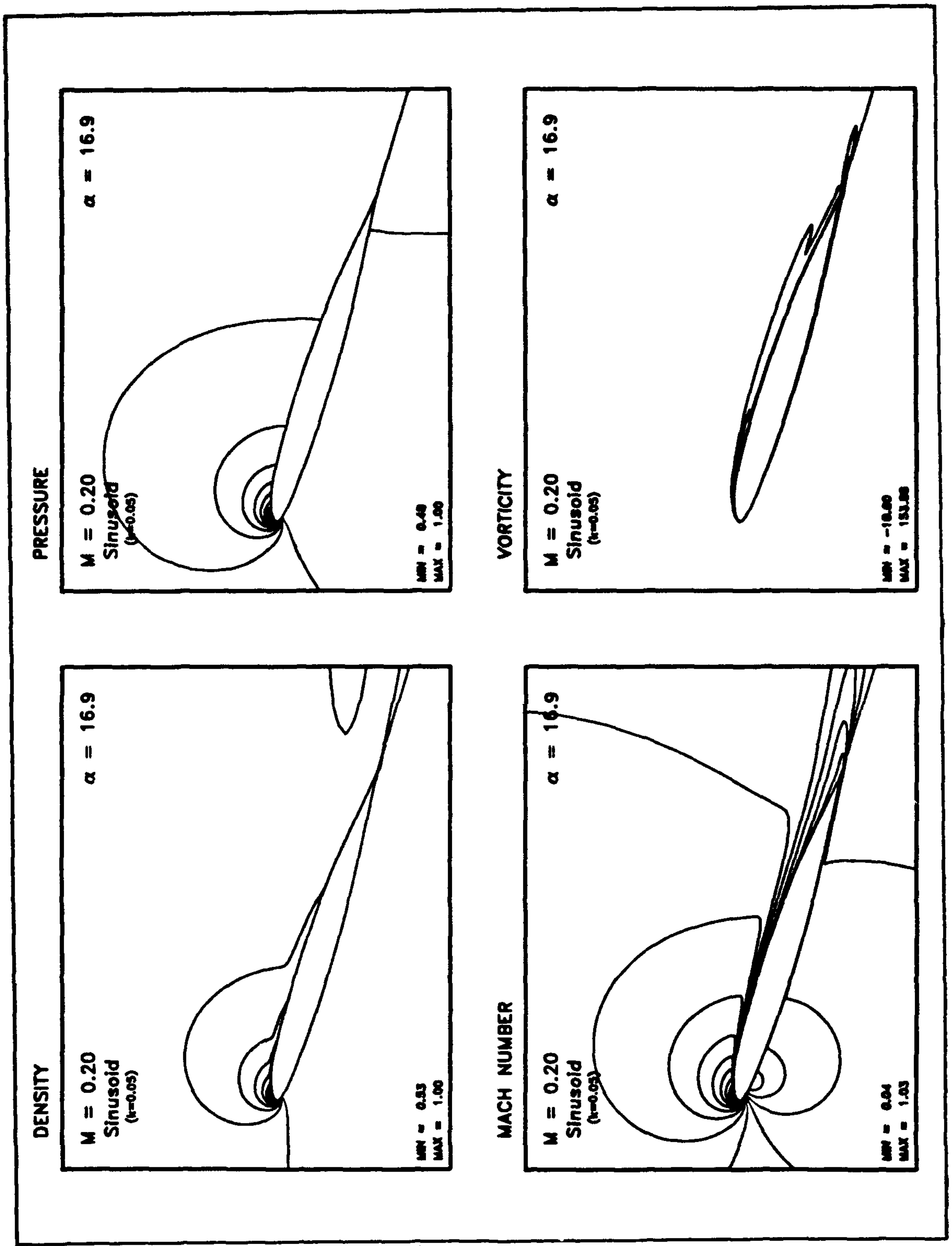


Figure 5.84
Sikorsky SSC-A09
Sinusoid (UP) (k=0.05, M=0.2)

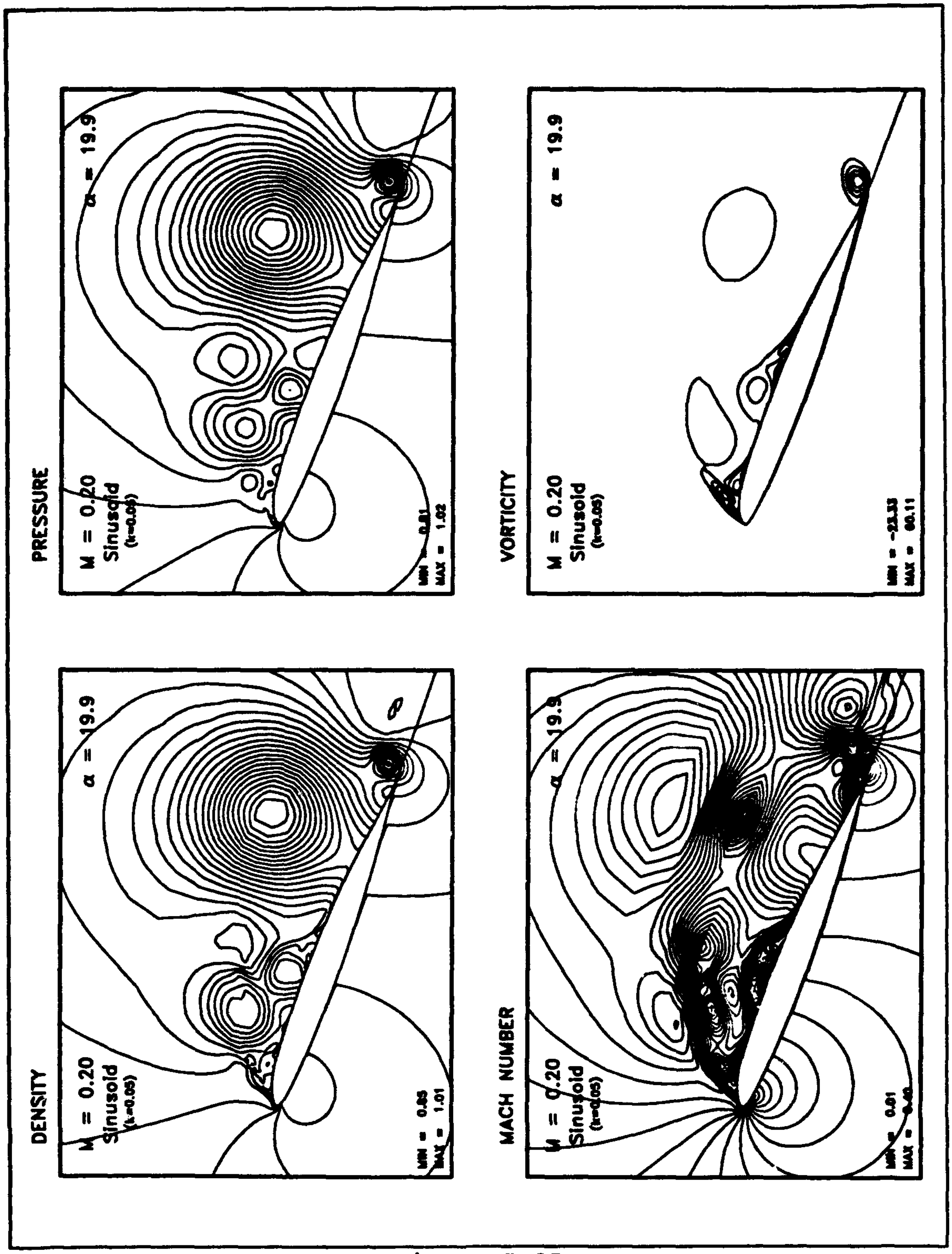


Figure 5.85
Sikorsky SSC-A09
Sinusoid (UP) (k=0.05, M=0.2)

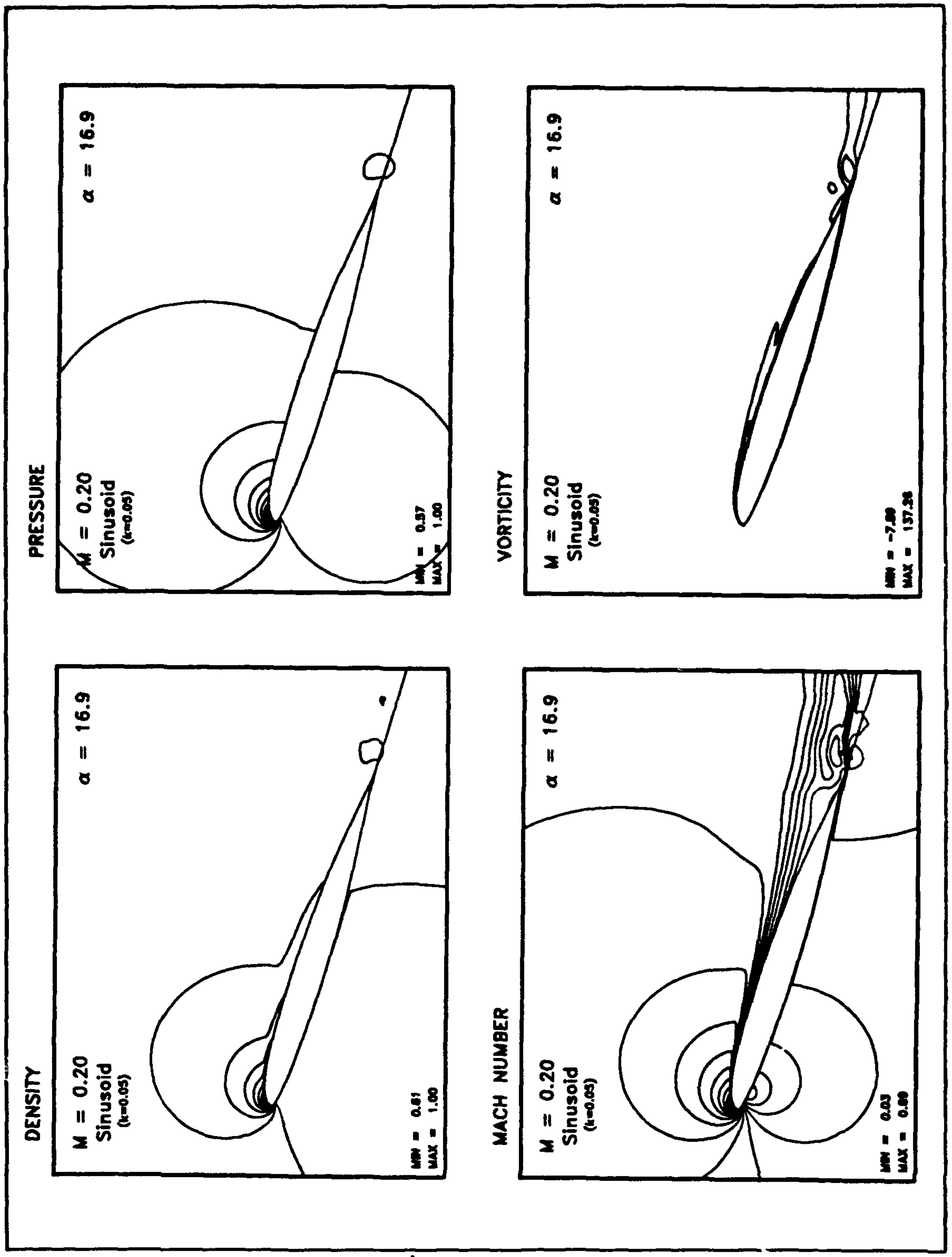


Figure 5.86
Sikorsky SSC-A09
Sinusoid (DOWN) (k=0.05, M=0.2)

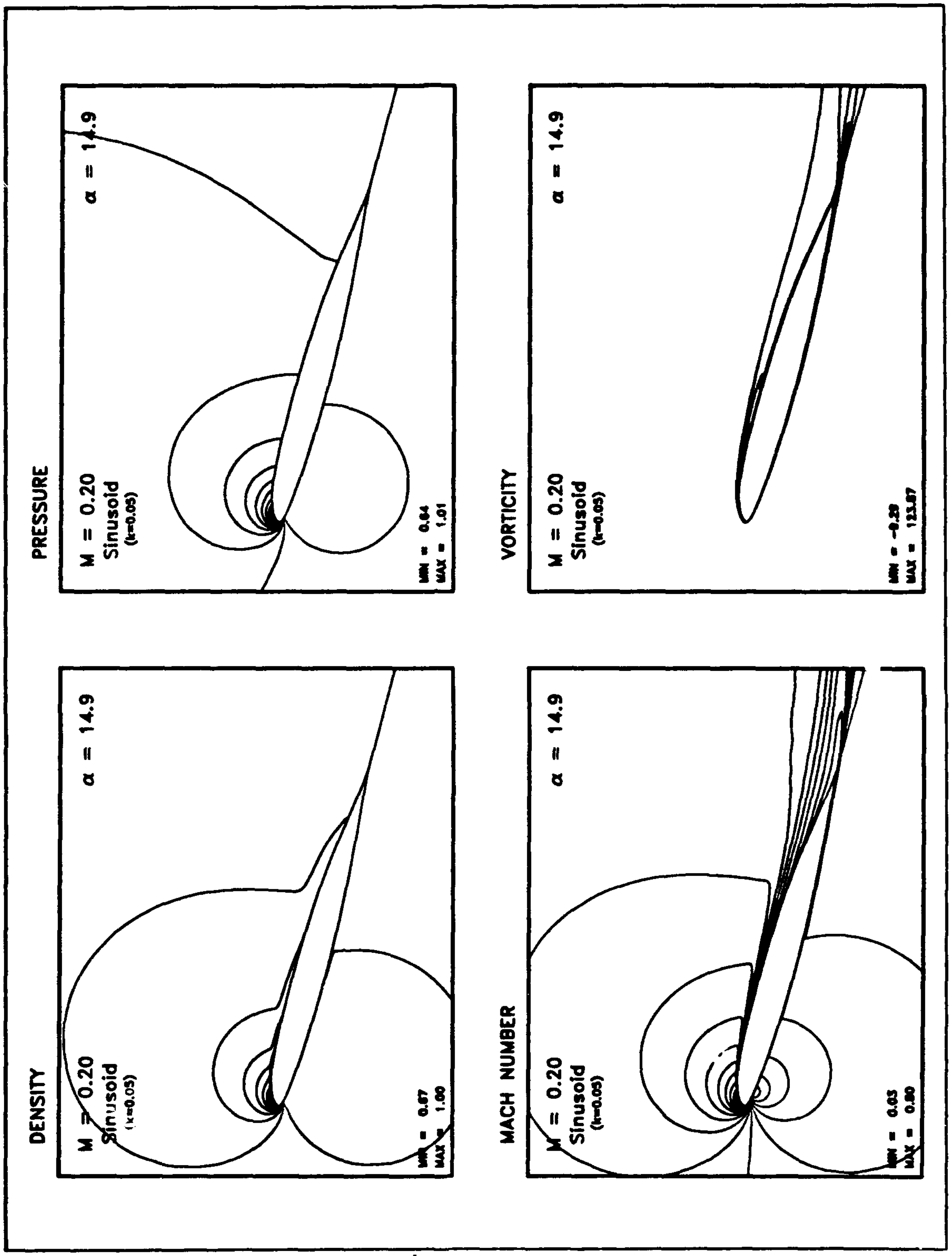


Figure 5.87
Sikorsky SSC-A09
Sinusoid (DOWN) (k=0.05, M=0.2)

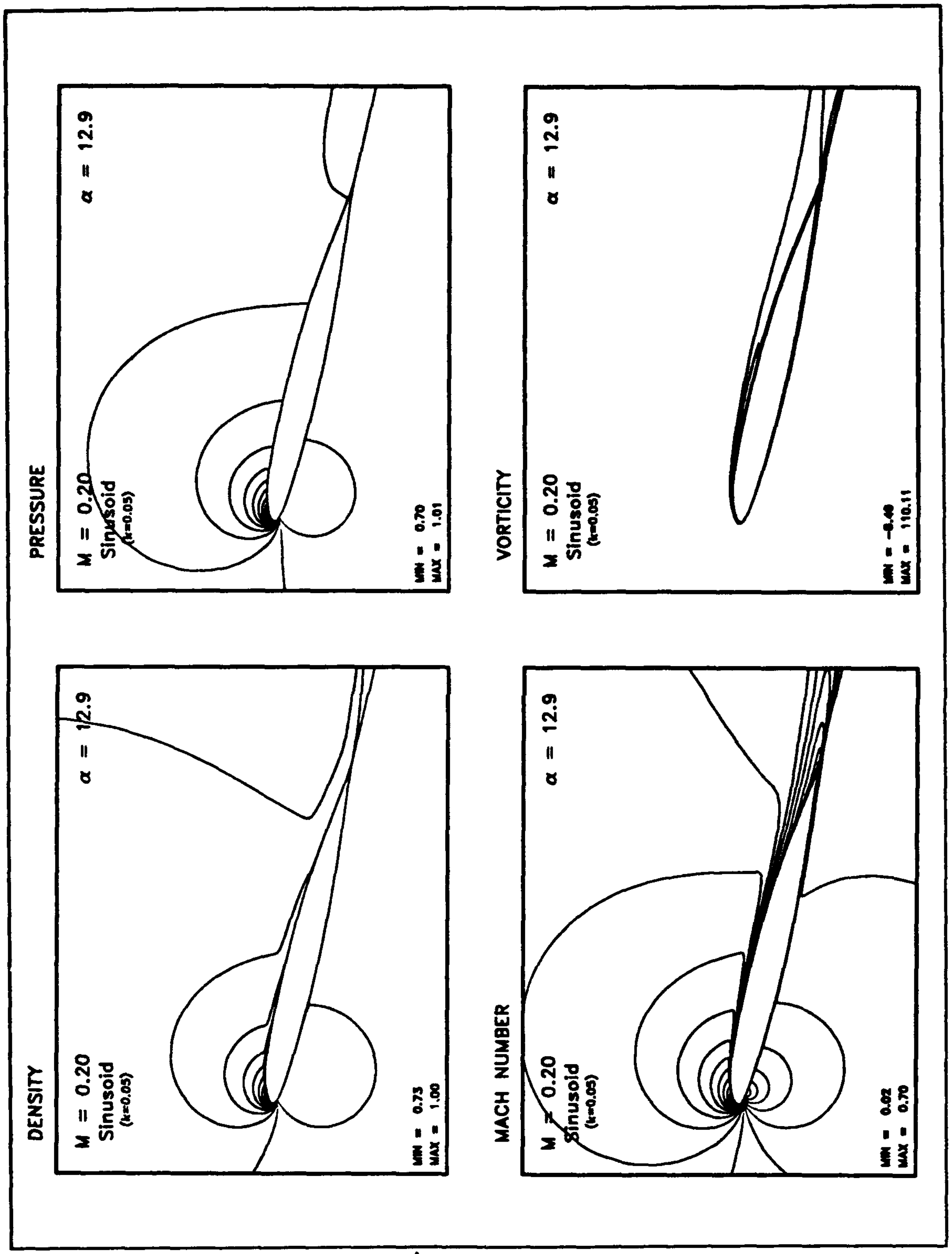


Figure 5.88
Sikorsky SSC-A09
Sinusoid (DOWN) (k=0.05, M=0.2)

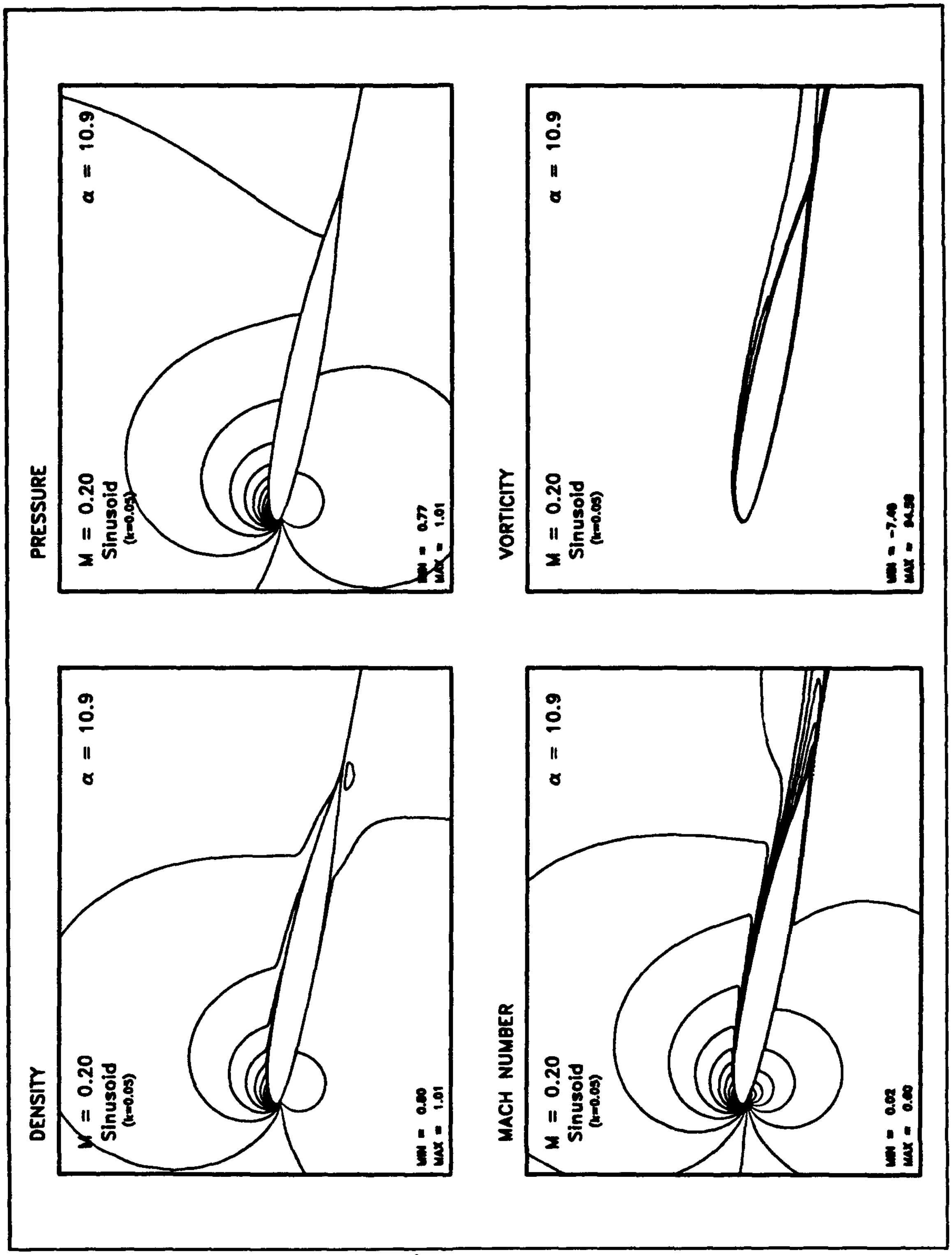


Figure 5.89
Sikorsky SSC-A09
Sinusoid (DOWN) (k=0.05, M=0.2)

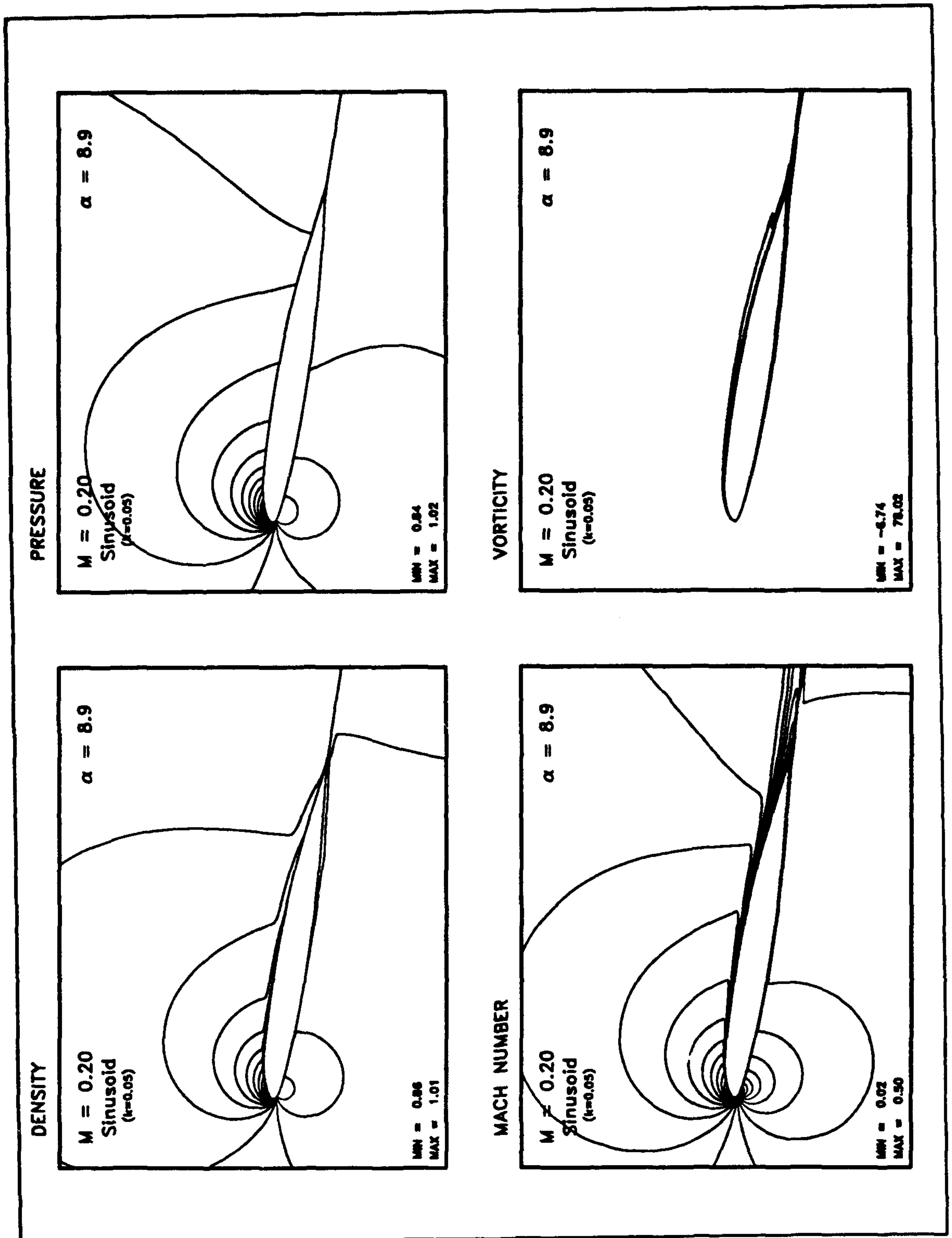


Figure 5.90
Sikorsky SSC-A09
Sinusoid (DOWN) (k=0.05, M=0.2)

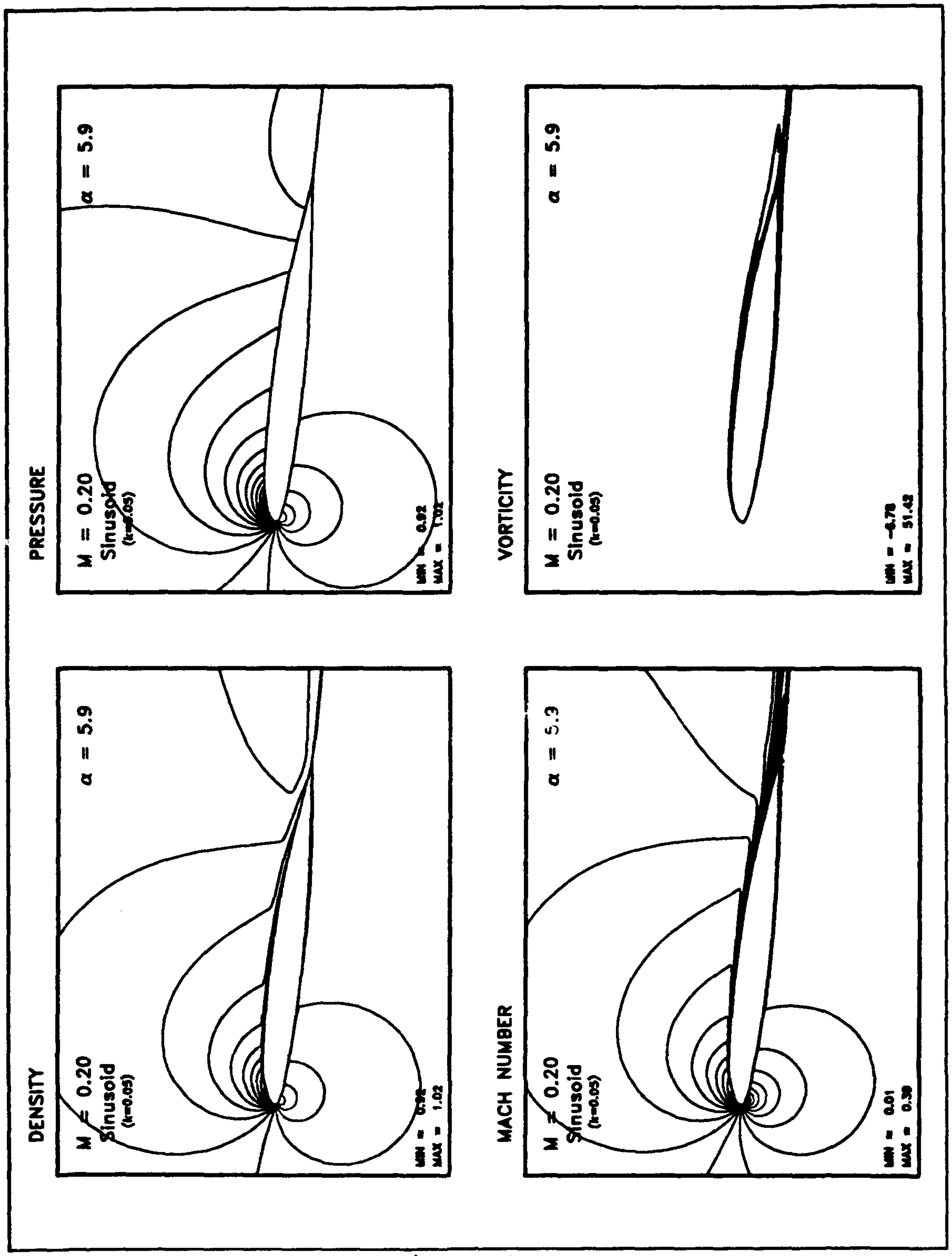


Figure 5.91
Sikorsky SSC-A09
Sinusoid (DOWN) (k=0.05, M=0.2)

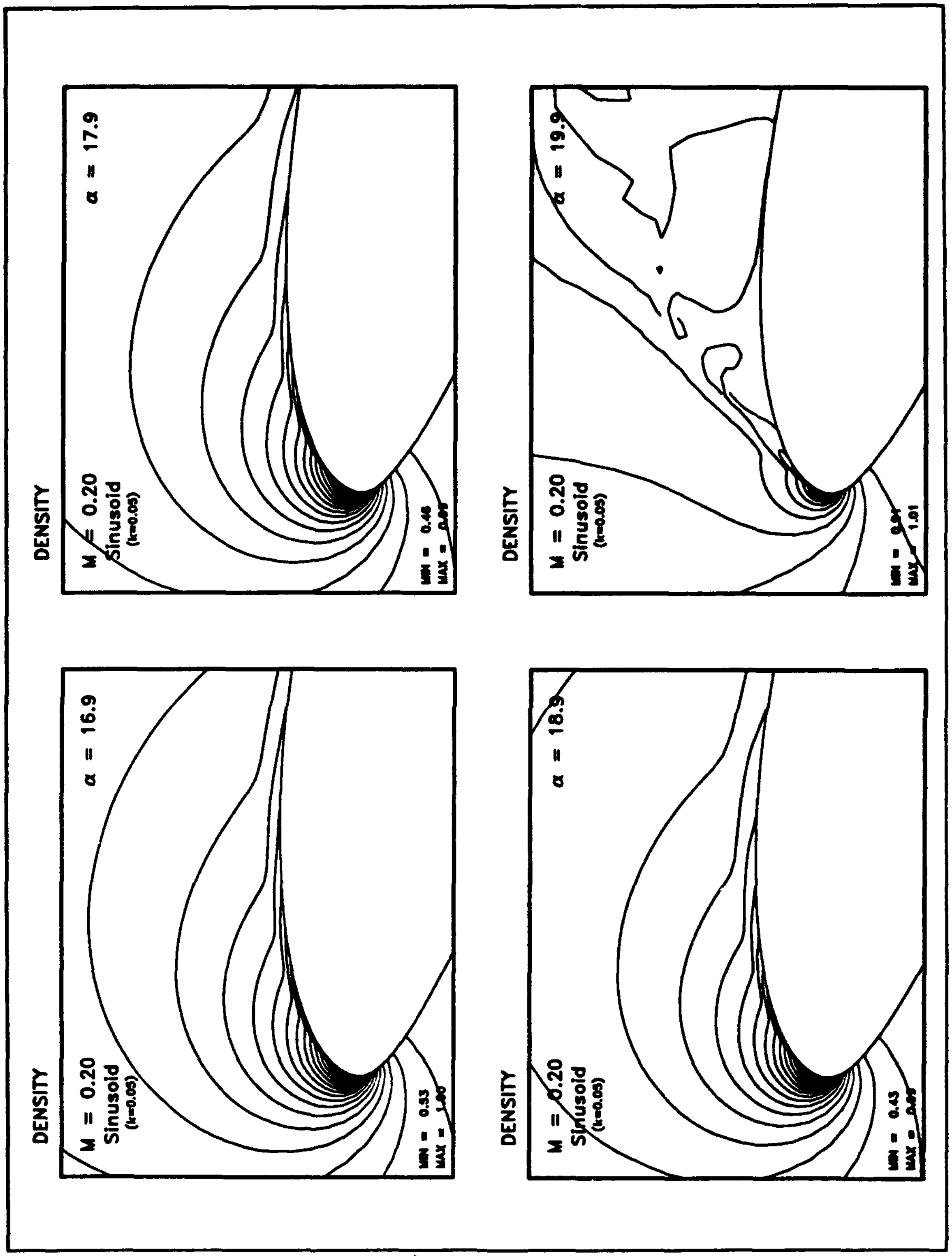


Figure 5.92
Sikorsky SSC-A09
Sinusoid (k=0.05, M=0.2)

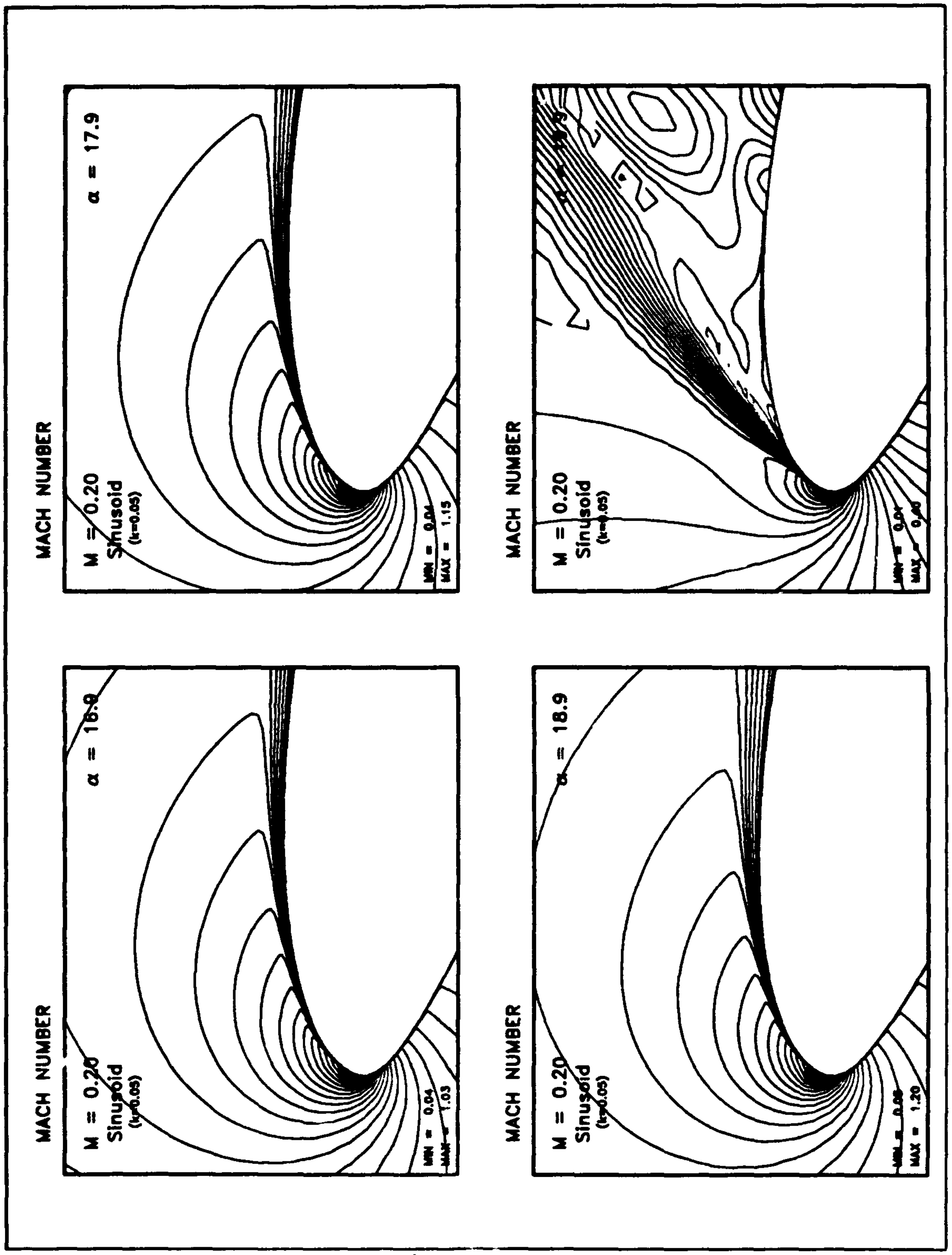


Figure 5.93
Sikorsky SSC-A09
Sinusoid (k=0.05, M=0.2)

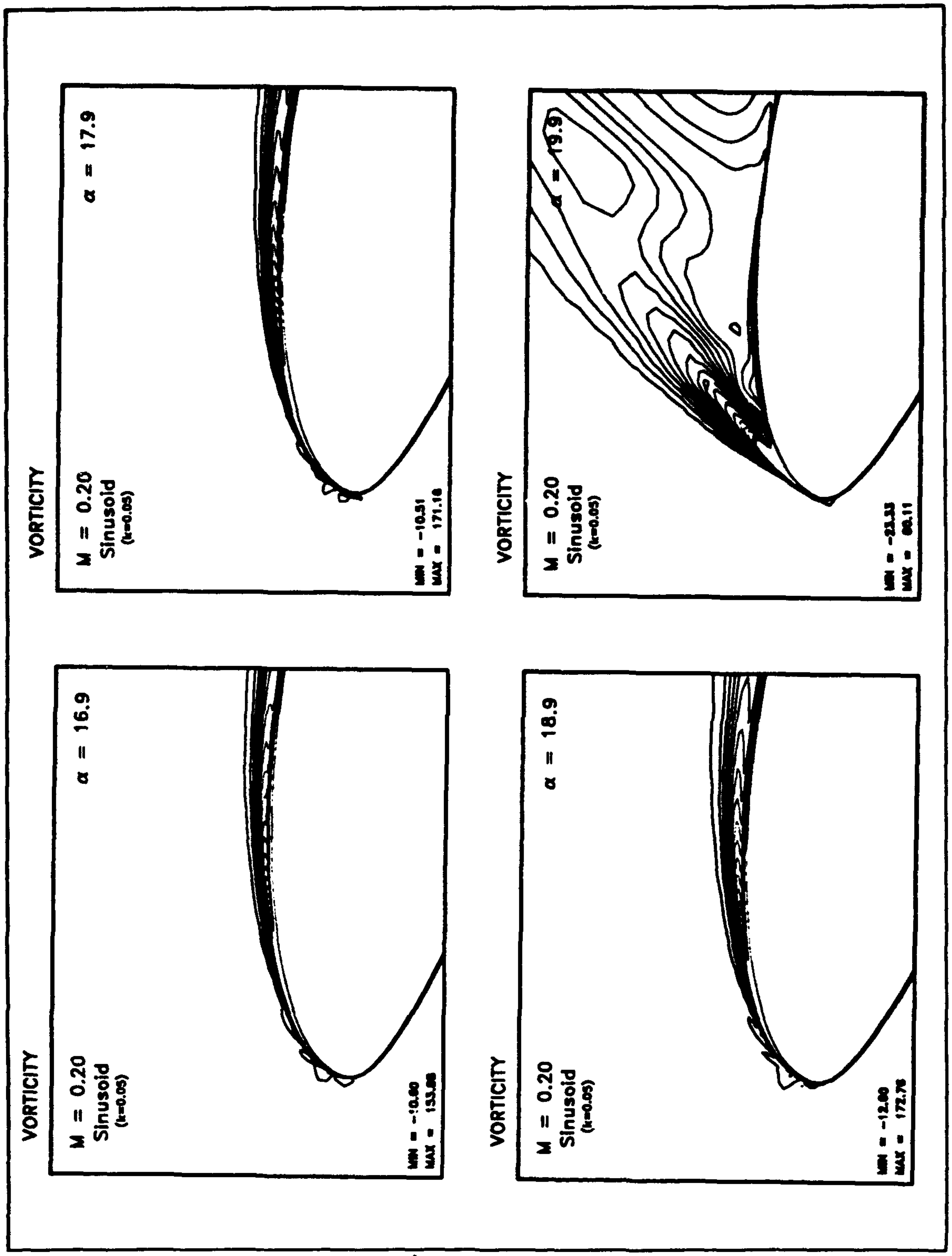


Figure 5.94
Sikorsky SSC-A09
Sinusoid (k=0.05, M=0.2)

VI. CONCLUSIONS AND RECOMMENDATIONS

The steady and unsteady two-dimensional flowfield analysis was conducted for a Sikorsky SSC-A09 airfoil in compressible, high Reynolds number flows. Computational methods included a steady panel method with compressibility corrections; a laminar and turbulent boundary layer method; an unsteady panel method; and a numerical solution method of the thin layer, compressible, Navier-Stokes equations. The Baldwin-Lomax, two-layer, zero-equation turbulence model was used. In steady flow with little or no separation, computed lift, drag, pitching moment, and skin friction coefficients, as well as displacement thickness and boundary layer velocity profiles at several angles-of-attack were generally found to be in good agreement with experimental data. The much simpler laminar and turbulent boundary layer method produced very good information, more than adequate for the preliminary design process, at a greatly reduced computational cost.

When any airfoil enters deep stall, the flowfield is seen to be dominated by massive flow separation and highly non-linear behavior characterized by the shedding of vortex-like structures. The Baldwin-Lomax, simple eddy viscosity turbulence model used here was found to be inadequate. It predicted steady flows accurately only when there was little or no flow separation. When the flow separated, it over

predicted the leading edge suction peak, over predicted experimentally shown separation, and consistently predicted higher lift and lower pitching-moment.

One major inaccuracy built into the assumptions was the transitional nature of the boundary layer being neglected and instead approximated by a fully turbulent flowfield. Here, the flowfield was shown to be dominated by leading edge separation, often induced by a shock. The small supersonic region and shock that form near the leading edge significantly reduced the peak suction pressure and airloads; thus negating the benefits from dynamic stall by reducing the stall vortex strength. The unsteady aerodynamic response near stall was shown to be strongly dependent on the leading edge stall vortex characteristics. Sinusoidal motions with higher reduced frequencies were shown to be qualitatively similar to ramp motion. Srinivasan, Ekaterinaris, and McCroskey [Ref. 14] concluded that no currently used turbulence model predicted all airloads consistently and in agreement with experiment for all flow conditions. They did conclude that the best improvement over the Baldwin-Lomax model was the Renormalization Group Theory of turbulence (RNG) model which also offered no additional computational costs. The RNG model is also an algebraic eddy viscosity, equilibrium model where the eddy viscosity is assumed to instantaneously adjust to the local flow without any history effects; and the specified integral length-scale is assumed proportional to the boundary

layer thickness. Srinivasan et al [Ref. 14] also concluded from a trade study that a denser C-type grid of 360 by 71 provided the best solution accuracy.

LIST OF REFERENCES

1. Anderson, D. A., Tannehill, J. C., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, Hemisphere Publishing Corporation, 1984.

2. Anderson, J. D. Jr, *Fundamentals of Aerodynamics*, second edition, Mcgraw-Hill Publishing Company, 1991.

3. Anderson, J. D. Jr, *Modern Compressible Flow*, second edition, Mcgraw-Hill Publishing Company, 1990.

4. Baldwin, B. S., Lomax, H., *Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows*, AIAA Paper 78-257, 1978.

5. Cebeci, T. and Bradshaw, P., *Momentum Transfer in Boundary Layers*, Hemisphere Publishing Corporation, 1977.

6. Cricelli, A., Ekaterinaris, J. A., and M. F. Platzer, *Unsteady Airfoil Flow Solutions on Moving Zonal Grids*, AIAA-92-0543, 1992.

7. Eppler, R., *Airfoil Design and Data*, Springer-Verlag, New York, 1990.

8. Harris, Charles D., *NASA Supercritical Airfoils*, NASA Technical Paper 2969, Langley Research Center, Hampton, Virginia, 1990.

9. Hess, J. L. and Smith, A. M., *Calculations of Potential Flow About Arbitrary Bodies*, Progress in Aeronautical Science, Vol 8, Pergamon Press, Oxford, 1966.

10. Kuethe, Arnold M. and Chow, Chuen-Yen, *Foundations of Aerodynamics: Bases of Aerodynamic Design*, third edition, John Wiley & Sons, 1976.

11. Lorber, P. F. and Carta, F. O., *Unsteady Stall Penetration Experiments at High Reynolds Number*, United Technologies Research Center, East Hartford, CT, April 1987.

12. **Merkle, C. L.,** *Computational Fluid Dynamics of Inviscid and High Reynolds Number Flows,* **First Edition, ProCopy Inc, State College, PA, 1990.**

13. **Nowak, L. M.,** *Computational Investigations of a NACA 0012 Airfoil in Low Reynolds Number Flows,* **Thesis, Naval Postgraduate School, Monterey, California, September 1992.**

14. **Srinivasan, G. R., Ekaterinaris, J. A., and McCroskey, W. J.,** *Dynamic Stall of an Oscillating Wing, Part 1: Evaluation of Turbulence Models,* **AIAA Paper 93-3403, 1993.**

15. **Teng, N. H., June 1987,** *The Development of a Computer Code (U2DIFF) for the Numerical Solution of Unsteady, Inviscid and Incompressible Flow over an Airfoil,* **Thesis, Naval Postgraduate School, Monterey, California.**

16. **Tuncer, I. H., Ekaterinaris, J. A., and Platzer, M. F., 1993,** *A Viscous-Inviscid Interaction Method for 2-D Unsteady, Compressible Flows,* **AIAA Paper 93-3019.**

17. **VanDyken, R. and Chandrasekhara, M., 1992,** *Leading Edge Velocity Field on an Oscillating Airfoil in Compressible Dynamic Stall,* **AIAA paper 92-0193.**

APPENDIX A

A. BASIC UNIX COMMANDS

man <topic> . . . . . . obtain help info on any 'topic'

yppasswd . . . . change your password

mkdir <name> . . . . make directory

cd <name> . . . . change directory to 'name'

ls . . . . list directory contents

ls -l . . w/permissions map and fn size (long option)

cp <fn1> <fn2> . copies 'fn1' to 'fn2' both in current directory

cp ../<fn1> . copy 'fn1' from above dir.Period specifies copy to same file name

mv <fn1> <fn2> . move/rename 'fn1' to 'fn2' & del fn1

rm <fn> . . . . remove 'fn'

rm -r <dir> . . removes all subfiles without prompting! no way to recover

pwd . . . . print working dir

more <fn> . . . . . . displays 'fn1' one page at a time

head +xx <fn> . . . . . . displays xx lines at top of fn

tail -xx <fn> . . . . displays xx lines at end of fn

df . . . . indicates disk free space

chmod +/- rwx <fn> . . . . change 'fn' protection

ps -ef . . . . show all sys processes (pid)

ps -ef |grep <name> . . . shows 'name' current processes

f77 -o <exe> <src.f> . . . compile fortran source code 'src.f' and use executable name 'exe'. -O2 or -O3 options are for vectorization

ctrl c . . . . kills a process

ctrl z . . . . suspends a process

lpstat -t . . . . shows all printing processes

cancel hp2p_ps - <pid> . . . . kills a print job

pa2ps <fn> . . . . . converts a 'fn' to postscript and prints two pages/sheet (NO GRAPHICS)

lp -dhp2p_ps <fn> . . . . prints a graphics file

diff <fn1> <fn2> . . . displays the difference between files 'fn1' and 'fn2'

zip <fn> . . . . windows type editor

***** . . . . Wild card

B. PROGRAM AIRFOIL.F

This program generates appropriately distributed airfoil surface coordinate points, N+1 points, for any NACA 4- or 5-Digit Airfoil and places them into 'points.dat' using the nodal spacing function:

$$.5 \times \left\{ 1 - \cos\left(\pi \times \frac{N_i}{N}\right) \right\}$$

1. Four Digit NACA Series (xxxx)

First digit: Maximum camber in hundredths of chord

Second digit: Location of maximum camber in tenths of chord

Third & 4^{th}: Maximum thickness in hundredths of chord

2. Five Digit NACA Series (xxxxx)

First digit: Multiplied by 3/2 is C_l in tenths

Second & 3^{rd}: Divided by 2 is location of maximum camber in hundredths of chord

4^{th} & 5^{th}: Maximum thickness in hundredths of chord

3. **Program Commands**

airfoil

"Input Desired Number of Panels on the Upper, Lower Surface." (max 100,100)

50,50

"Input NACA 4- or 5- Digit Airfoil Type."

XXXX or **XXXXX**

Note: Surface Coordinates are created and stored into 'points.dat' where {#panels = #points - 1}.
If desired, create or copy your own surface coordinates to 'points.dat'

c. PROGRAM PANEL.F

This program reads in 'Points.dat', calculates Velocity and Pressure distributions, and outputs BL2D.F required input data to 'bl2d.dat'. It generates the files: 'vel.dat', 'cp.dat', 'bl2d.dat', and 'cldm.dat'. It then calculates and displays force coefficient outputs c_l, c_d, and c_m.

1. **Program Commands**

panel

"Input the # Panels" (one less than in 'Points.dat')
(sum of two 'airfoil.f' panel entries)

100

"Input R_e (xE6)"(1 → 1,000,000)

1

"Enter Transition Location"

0 → Unknown

1 → Known

{If '1' chosen}

"Input X/C Transition Location Upper Surface"

(Note: The most CRITICAL input !)

0.XX

"Input X/C Transition Location Lower Surface"

0.XX

"Input Angle-of-Attack in Degrees"

X or **XX.X**

"Input Mach Number"

0 → Incompressible, or

.X → Compressible

D. PROGRAM BL2D.F

This program reads in 'bl2d.dat' and checks for proper stagnation point location. It then generates the files 'bl2d.out', 'cf.dat', 'dls.dat', 'pro1.dat', and 'pro2.dat'.

1. Program Commands

bl2d

"Reading in the data"

"I stagnation is = xx"

"Input new I stagnation ="

xx

"Boundary layer computation in progress"

"Boundary layer computation in progress"

"Michels Transition estimates"

"Estimate for upper transition = .xx"

"Estimate for lower transition = .xx"

2. **Program optimization**

The program is repeatedly run until convergence is obtained. Procedures:

- Run bl2d
- Run gnuplot
- Load 'dis' and check C_F for convergence
- If diverged solution, **change the upper transition point** (forward) or alter the stagnation point (±1) and run bl2d again.
- Repeat the above until convergence is reached.
- Move the transition point progressively aft. Find the point where BL2D will just converge. **This is the Transition Point.**

E. VISUALIZATION ROUTINES

Data output is viewed using Gnuplot batch files. Gnuplot is a plotting routine available on the Indigos. Procedures to view or print your results:

g	Activates gnuplot plotting routine
l 'press'	View Pressure Distribution (l = load)
l 'vel'	View Velocity distribution
l 'cldm'	View $C_{l\alpha}$, $C_{d\alpha}$, or $C_{m\alpha}$
l 'dis'	View C_F and Displacement Thickness
l 'profile'	View Boundary Layer Velocity Profiles
l 'pt'	Prints any graph displayed on your screen to the hp2p_ps printer

1. Gnuplot Commands

set autoscale . . . . autoscale all axis
set title ' ' . . . . graph title
set ylabel ' ' . . . . y title
set xlabel ' ' . . . . x title
set xrange [:]
set yrange [:]
set grid . . . . turn on grid
set nogrid . . . . turn off grid
set key x,y . . . . moves legend to x,y
set nokey . . . . no legend
replot . . . . redisplays plot with new changes
set label '' at x,y displays a label at position x,y
set data style points . . . displays data as points
set data style lines . . . . displays data as lines
set data style linespoints . displays data as lines & points
p 'cp.dat' w lines . . . plots 'cp.dat' using lines
p 'cldm.dat' u 1:3 w lines . plots 'cldm.dat' using 1:3 (α vs C_d) using lines

2. Gnuplot Batch Files

PRESS

```
#Cp distribution
set grid
set key
set label
set function style lines
set tics out
set ticslevel 0.5
set xtics
set ytics
set ztics
set title "Pressure Coefficient Distribution" 0,1
set xlabel "X/C" 0,-1
set xrange [0 : 1]
set ylabel " Cp " 0,.5
set yrange [1 : 4]
plot 'cp.dat' w lines
```

VEL

```
#Velocity distribution
set grid
set key
set label
set function style lines
set tics out
set ticslevel 0.5
set xtics
set ytics
set ztics
set title "Velocity Distribution" 0,1
set xlabel "X/C" 0,-1
set xrange [0 : 1]
set ylabel " Velocity " 0,.5
set yrange [0 : 4]
plot 'vel.dat' w lines
```

CLDM

```
#cldm distribution
set grid
set key
set label
set function style lines
set tics out
set ticslevel 0.5
set xtics
set ytics
set ztics
set autoscale
set title "Lift Coefficient Distribution" 0,1
set xlabel "AOA" 0,-1
#set xrange [0 : 1]
set ylabel "Cl" 0,.5
#set yrange [1 : 4]
#plot 'cldm0012.dat' u 1:3, 'cldm2412.dat' u 1:3
plot 'cldm.dat' u 1:2
```

DIS

```
# Cf and Del* Distribution
set grid
set key
set label
set border
set function style lines
set tics out
set ticslevel 0.5
set xtics
set ytics -.02,.001,.01
set ztics
set title "Cf and Delta-Star" 0,1
set xlabel "X/C" 0,-1
```

```
set xrange [0:1]
set yrange [-.002 : .006]
plot "cf.dat" w lines,  "dls.dat" w lines
```

PROFILE

```
# Boundary Layer Velocity Profiles
#set terminal x11
set grid
set key
set nolabel
set function style lines
set tics in
set ticslevel 0.5
set xtics
set ytics
set ztics
set title "Boundary Layer Velocity Profiles" 0,0
set xlabel "Airfoil Upper Surface" 0,0
set xrange [0 : 10]
set ylabel "y/c" 0,.5
set yrange [0 : .0015]
#plot "pro14.x25" w lines, "pro24.x25" w lines
plot "pro1.dat" w lines, "pro2.dat" w lines
```

PT

```
# Plotting routine for HP2P postscript printer
set term postscript
set output 'pt.gr'
replot
set term x11
!lp -dhp2p_ps pt.gr
```

F. AIRFOIL.F & PANEL.F & BL2D.F SOURCE CODES

PROGRAM AIRFOIL

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
C      CDR T. Johnston     (AUGUST 1993)
C         Generates surface coordinates for any 4 or 5 digit NACA airfoil
C      and puts them in 'points.dat' using spacing function:
C                           .5 * ( 1. - cos(PI* (node/total points) ) )
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
       COMMON /BOD/ NLOWER,NUPPER,NODTOT,X(202),Y(202)                  U2D00190
       COMMON /PAR/ NACA,TAU,EPSMAX,PTMAX                               U2D03350
       OPEN(unit=3,file='points.dat',status='unknown')
       CALL INDATA
       CALL SETUP
       WRITE (3,1010) (X(I),Y(I),I=1,NODTOT+1)                          U2D00470
       WRITE (6,1010) (X(I),Y(I),I=1,NODTOT+1)
 1010  FORMAT (2(F8.6,2x))                                              U2D00480
       end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccU2D03220
C      SUBROUTINE BODY(Z,SIGN,X,Y)                                     CU2D03240
C               RETURN COORDINATES OF POINT ON THE BODY SURFACE        CU2D03260
C                      Z = NODE-SPACING PARAMETER                      CU2D03280
C                      X,Y = CARTESIAN COORDINATES                     CU2D03290
C                      SIGN = +1. FOR UPPER SURFACE                    CU2D03300
C                             -1. FOR LOWER SURFACE                    CU2D03310
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccU2D03330
       SUBROUTINE BODY(Z,SIGN,X,Y)                                      U2D03340
       COMMON /PAR/ NACA,TAU,EPSMAX,PTMAX                               U2D03350
       IF (SIGN .LT. 0.0)     Z = 1. - Z                                U2D03360
       CALL NACA45(Z,THICK,CAMBER,BETA)                                 U2D03370
       X       = Z - SIGN*THICK*SIN(BETA)                               U2D03380
       Y       = CAMBER + SIGN*THICK*COS(BETA)                          U2D03390
       RETURN                                                           U2D03400
       END                                                              U2D03410
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccU2D06440
C      SUBROUTINE INDATA                                               CU2D06460
C                     SET PARAMETERS OF BODY SHAPE                     CU2D06480
C                     FLOW SITUATION, AND NODE DISTRIBUTION            CU2D06490
C                     USER MUST INPUT                                  CU2D06510
C                          NLOWER = NUMBER OF NODES ON LOWER SURFACE   CU2D06520
C                          NUPPER = NUMBER OF NODES ON UPPER SURFACE   CU2D06530
C                     PLUS DATA ON BODY AND SUBROUTINE BODY            CU2D06540
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCU2D06560
      SUBROUTINE INDATA                                                 U2D06570
      COMMON /BOD/ NLOWER,NUPPER,NODTOT,X(202),Y(202)                   U2D06590
      COMMON /PAR/ NACA,TAU,EPSMAX,PTMAX                                U2D06610
      WRITE(6,*) 'INPUT THE NUMBER OF PANELS ON UPPER AND LOWER SUFACE:
     +NLOWER, NUPPER ?'
      READ (5,*) NLOWER,NUPPER                                          U2D06700
      WRITE (6,558)NLOWER,NUPPER                                        U2D06710
  558 FORMAT (///2X, '========================================',/,
     2        2X, 'NO. PANELS UPPER SURFACE     ='        ,I5,/,
     3        2X, 'NO. PANELS LOWER SURFACE     ='        ,I5,/,
     4        2X, '========================================')
      WRITE(6,*) 'INPUT THE NACA AIRFOIL TYPE: 4 or 5 digit series (IE.
     +XXXX, or 230XX) ?'
      READ (5,*) NACA                                                   U2D06730
      WRITE (6,*) NACA                                                  U2D06740
      IEPS    = NACA/1000                                               U2D06750
      IPTMAX  = NACA/100 - 10*IEPS                                      U2D06760
      ITAU    = NACA - 1000*IEPS - 100*IPTMAX                           U2D06770
      EPSMAX  = IEPS*0.01                                               U2D06780
      PTMAX   = IPTMAX*0.1                                              U2D06790
      TAU     = ITAU*0.01                                               U2D06800
      IF (IEPS .LT. 10) RETURN                                          U2D06810
      PTMAX   = 0.2025                                                  U2D06820
      EPSMAX  = 2.6595*PTMAX**3                                         U2D06830
      RETURN                                                            U2D06840
      END                                                               U2D06850
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCU2D09120
C     SUBROUTINE NACA45(Z,THICK,CAMBER,BETA)                           CU2D09140
C             EVALUATE THICKNESS AND CAMBER                            CU2D09160
C             FOR NACA 4- OR 5-DIGIT AIRFOIL                           CU2D09170
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCU2D09190
      SUBROUTINE NACA45(Z,THICK,CAMBER,BETA)                            U2D09200
      COMMON /PAR/ NACA,TAU,EPSMAX,PTMAX                                U2D09210
      THICK   = 0.0                                                     U2D09220
      IF (Z .LT. 1.E-10)     GO TO 100                                  U2D09230
      THICK   = 5.*TAU*(.2969*SQRT(Z) - Z*(.126 + Z*(.3537              U2D09240
     +          - Z*(.2843 - Z*.1015))))                                U2D09250
  100 IF (EPSMAX .EQ. 0.0) GO TO 130                                    U2D09260
      IF (NACA .GT. 9999) GO TO 140                                     U2D09270
      IF (Z .GT. PTMAX) GO TO 110                                       U2D09280
      CAMBER  = EPSMAX/PTMAX/PTMAX*(2.*PTMAX - Z)*Z                     U2D09290
      DCAMDX  = 2.*EPSMAX/PTMAX/PTMAX*(PTMAX - Z)                       U2D09300
      GO TO 120                                                         U2D09310
  110 CAMBER  = EPSMAX/(1.-PTMAX)**2*(1. + Z - 2.*PTMAX)*(1. - Z)       U2D09320
      DCAMDX  = 2.*EPSMAX/(1.-PTMAX)**2*(PTMAX - Z)                     U2D09330
  120 BETA    = ATAN(DCAMDX)                                            U2D09340
      RETURN                                                            U2D09350
  130 CAMBER  = 0.0                                                     U2D09360
      BETA    = 0.0                                                     U2D09370
      RETURN                                                            U2D09380
  140 IF (Z .GT. PTMAX)      GO TO 150                                  U2D09390
      W       = Z/PTMAX                                                 U2D09400
      CAMBER  = EPSMAX*W*((W - 3.)*W + 3. - PTMAX)                      U2D09410
      DCAMDX  = EPSMAX*3.*W*(1. - W)/PTMAX                              U2D09420
      GO TO 120                                                         U2D09430
  150 CAMBER  = EPSMAX*(1. - Z)                                         U2D09440
      DCAMDX  = - EPSMAX                                                U2D09450
      GO TO 120                                                         U2D09460
      END                                                               U2D09470
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCU2D11360
C     SUBROUTINE SETUP                                                 CU2D11380
C             SET-UP COORDINATES OF PANEL NODES AND SLOPES OF PANELS   CU2D11400
C             COORDINATES ARE READ FROM INPUT DATA FILE UNLESS         CU2D11410
C                  THE AIRFOIL IS OF NACA XXXX OR NACA 230XX TYPE      CU2D11420
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCU2D11440
      SUBROUTINE SETUP                                                  U2D11450
      COMMON /BOD/ NLOWER,NUPPER,NODTOT,X(202),Y(202)                   U2D11460
      PI      = 3.1415926585                                            U2D11490
      PI2INV  = .5/PI                                                   U2D11500
C             SET COORDINATES OF NODES ON BODY SURFACE                  U2D11520
      NPOINT  = NLOWER                                                  U2D11550
      SIGN    = -1.0                                                    U2D11560
      NSTART  = 0                                                       U2D11570
      DO 110  NSURF = 1,2                                               U2D11580
      DO 100  N = 1,NPOINT                                              U2D11590
      FRACT   = FLOAT(N-1)/FLOAT(NPOINT)                                U2D11600
      Z       = .5*(1. - COS(PI*FRACT))                                 U2D11610
```

```
      I        = NSTART + N                                           U2D11620
      CALL BODY(Z,SIGN,X(I),Y(I))                                     U2D11630
  100 CONTINUE                                                        U2D11640
      NPOINT   = NUPPER                                               U2D11650
      SIGN     = 1.0                                                  U2D11660
      NSTART   = NLOWER                                               U2D11670
  110 CONTINUE                                                        U2D11680
      NODTOT   = NLOWER + NUPPER                                      U2D11690
      X(NODTOT+1) = X(1)                                              U2D11700
      Y(NODTOT+1) = Y(1)                                              U2D11710
      RETURN                                                          U2D11930
      END                                                             U2D11940
************************************************************************
************************************************************************
```

PROGRAM PANEL

```
*********************************************************************
* *****                    AUGUST 1993                          ****
*                  +++++    CDR T. Johnston    +++++
* PURPOSE: CALCULATE THE VELOCITIES ON AN AIRFOIL USING A PANEL METHOD.
*     LIM: Arrays currently dimensioned for maximum of N=200 panels
*          Input data file points.dat will have N+1 points
*          Output velocities are referenced to freestream, ie. V/Vinf

*  METHOD: FLOWFIELD CONSISTS OF THREE SIMPLER FLOWS:  FREESTREAM, SOURCE,
*          AND VORTICITY.  SOURCE DISTRIBUTIONS q(j) VARY FROM PANEL TO
*          PANEL.  VORTICITY STRENGTH GAMMA IS THE SAME FOR ALL PANELS.
*          BOUNDARY CONDITIONS INCLUDE FLOW TANGENCY AT CONTROL POINTS AND
*          KUTTA CONDITION FOR FIRST AND LAST PANELS.  INFLUENCE
*          COEFFICIENTS COMBINED TO FORM NEW COEFFICIENTS IN LINEAR SYSTEM
*          OF n+1 EQUATIONS, n+1 UNKNOWNS (q(1)...q(n), GAMMA).  VELOCITIES
*          AT CONTROL POINTS EVALUATED FROM q(j) AND GAMMA.
************************************************************************
*     Variable Definitions
*    N          = # Panels
*    RL         = Reynolds Number
*    IANS       = USER Transition Location Flag
*    TRANSUPPER = Upper Transition Point
*    TRANSLOWER = Lower Transition Point
*    ALPHA      = Angle of Attack
*    P          = Mach Number
*    BETA1      = SQRT ( 1-M^2 )
*    X(I)       = X Surface Coordinates
*    Y(I)       = Y Surface Coordinates
*    XM(I)      = X Control Points
*    YM(I)      = Y Control Points
*    R(I,J)     = Distance, Control Point to Panel with Unit Strength
*                   Singularity Distribution
*    AN(I,J)    = Normal Velocity Component Induced at Ith Panel Control
*                   Point by a Unit Source on Jth Panel
*    AT(I,J)    = Tangential Velocity Component Induced at Ith Panel Control
*                   Point by a Unit Source on Jth Panel
*    BN(I,J)    = Normal Velocity Component Induced at Ith Panel Control
*                   Point by a Unit Vorticity on Jth Panel
*    BT(I,J)    = Tangential Velocity Component Induced at Ith Panel Control
*                   Point by a Unit Vorticity on Jth Panel
*    q(I)       = Source Strength
*    GAMMA(I)   = Vorticity Strength
*    Vt(I)      = Normalized Tangential Velocity @ I Control Point ( V/Vinf )
*    Vtc(I)     = Compressible Normalized Tangential Velocity @ I Control pt
*    CP         = Pressure Coefficient     ( Cp = 1 - Vt^2 )
*    ISTAG      = Location of Stagnation Panel (Vt reversal)
************************************************************************
       REAL      X(1:202),Y(1:202),XM(1:202),YM(1:202),
     :           At(1:202,1:202),Bt(1:202,1:202),
     :           a(1:202,1:202),b(1:202),
     :           q(1:202),Vt(1:202),ALPHA,VtC(1:202),cP,cPc,CCPC(202),
     :           PI,GAMMA,THETA(1:202),NUM,DEN,
     :           R(1:202,1:202),BETA(1:202,1:202),NUM1,DEN1,NUM2,DEN2,
     :           AAUG(1:202,1:202),An(1:202,1:202),Bn(1:202,1:202)

      * NUMBER OF PANELS ON AIRFOIL SURFACE:
        PRINT*,'INPUT NO. OF PANELS (1 less than #lines in points.dat):'
        READ *,N
       PI=ACOS(-1.)
```

```
          OPEN (UNIT=88,FILE='points.dat',STATUS='UNKNOWN')
          OPEN (UNIT=89,FILE='vel.dat',STATUS='UNKNOWN')
          OPEN (UNIT=81,FILE='cp.dat',STATUS='UNKNOWN')
          OPEN (UNIT=90,FILE='bl2d.dat',STATUS='UNKNOWN')
          OPEN (UNIT=91,FILE='cldm.dat',STATUS='UNKNOWN')
          print *,'INPUT REYNOLDS NUMBER = ____ (X  E+6)'
          READ *,RL
          RL = RL*1000000.

          print *,'ENTER 0 IF TRANSITION LOCATIONS UNKNOWN (.8 & .999 USED)'
          PRINT *,'       1 IF You wish to enter Transition Locations'
          READ *,IANS

          IF(IANS.EQ.1) THEN
          PRINT *,'INPUT X/C TRANSITION LOCATION FOR UPPER SURFACE:'
          READ *, TRANSUPPER
          PRINT *,'INPUT X/C TRANSITION LOCATION FOR LOWER SURFACE:'
          READ *, TRANSLOWER
          ELSE
***These are arbitrary values intended to be downstream of the
***  actual transition points, for use with Michel's criterion in BL2D
          TRANSUPPER=.8
          TRANSLOWER=.999
          ENDIF
          WRITE (90,50) RL,TRANSUPPER,TRANSLOWER
50        FORMAT (F10.0,F10.6,F10.6)
          PRINT *,'INPUT ANGLE OF ATTACK IN DEGREES:'
          READ *,ALPHA
          ALPHA=ALPHA*PI/180.0
          PRINT *,'INPUT MACH NUMBER (0 FOR INCOMPRESSIBLE):'
          READ *,P
          p = rl/10000000.
          BETA1=(1.0-P**2.0)**.5

**        Read in Points.dat - Surface Coordinates must be TE, Clockwise,TE
         DO 30 I=1,N+1
         READ (88,*) X(I),Y(I)
30        CONTINUE

*         This section defines the influence coefficients:
          DO 110 I=1,N
*           Control Points
         XM(I)=0.5*(X(I)+X(I+1))
         YM(I)=0.5*(Y(I)+Y(I+1))
         R(I,1)=(XM(I)-X(1))**2.+(YM(I)-Y(1))**2.
         DO 100 J=1,N
           NUM=Y(J+1)-Y(J)
           DEN=X(J+1)-X(J)
           THETA(J)=ATAN2(NUM,DEN)
           NUM1=YM(I)-Y(J+1)
           DEN1=XM(I)-X(J+1)
           NUM2=YM(I)-Y(J)
           DEN2=XM(I)-X(J)
           BETA(I,J)=ATAN2((NUM1*DEN2-DEN1*NUM2),(DEN1*DEN2+NUM1*NUM2))
           R(I,J+1)=(XM(I)-X(J+1))**2.+(YM(I)-Y(J+1))**2.
           THETADIF=THETA(I)-THETA(J)
           IF (I.EQ.J)
     :          THEN
               An(I,J)=0.5
               Bn(I,J)=0.0
             ELSE
              An(I,J)=(1/(2*PI))*(SIN(THETADIF)*LOG(R(I,J+1)/R(I,J))
     :                     *.5+COS(THETADIF)*BETA(I,J))
              Bn(I,J)=(1/(2*PI))*(COS(THETADIF)*LOG(R(I,J+1)/R(I,J))
     :                     *.5-SIN(THETADIF)*BETA(I,J))
           END IF
           At(I,J)=-Bn(I,J)
           Bt(I,J)=An(I,J)
100         CONTINUE
110       CONTINUE

* Matrix coefficients of linear system defined (a's and b's):
          a(N+1,N+1)=0.0
         DO 130 I=1,N
          a(I,N+1)=0.0
         DO 120 J=1,N
           a(I,J)=An(I,J)
           a(I,N+1)=a(I,N+1)+Bn(I,J)
```

```
      120      CONTINUE
          b(I)=-1.0*SIN(ALPHA-THETA(I))
          a(N+1,I)=At(1,I)+At(N,I)
          a(N+1,N+1)=a(N+1,N+1)+Bt(1,I)+Bt(N,I)
      130     CONTINUE
           b(N+1)=-1.0*(COS(ALPHA-THETA(1))+COS(ALPHA-THETA(N)))

      * Define augmented matrix for input to linear solver subroutine GAUSS
          DO 150 I=1,N+1
           DO 140 J=1,N+1
             AAUG(I,J)=a(I,J)
      140      CONTINUE
            AAUG(I,N+2)=b(I)
      150    CONTINUE

          CALL GAUSS(N+1,AAUG)

      * Define source and vorticity strengths:
           DO 160 I=1,N
          q(I)=AAUG(I,N+2)
      160    CONTINUE
           GAMMA=AAUG(N+1,N+2)

      * Calculate velocity on each panel at control point
           NSTAGFLAG=0
           ISTAG=0
           DO 180 I=1,N
          Vt(I)=0.0
          DO 170 J=1,N
            Vt(I)=At(I,J)*q(J)+GAMMA*Bt(I,J)+Vt(I)
      170      CONTINUE
          Vt(I)=Vt(I)+COS(ALPHA-THETA(I))
              VtC(I)=Vt(I)/BETA1
          Cp=1.0-Vt(I)**2
              CpC=1.0-VtC(I)**2

      **       Find Stagnation Point for BL2D2
          IF ((Vt(I).GT.0) .AND. (NSTAGFLAG.EQ.0)) THEN
             ISTAG=I
             NSTAGFLAG=1
          ENDIF
      **       Output only Positive Velocities
          IF (Vt(I).LT.0) Vt(I)=-Vt(I)
              IF (VtC(I).LT.0) VtC(I)=-VtC(I)
          WRITE (89,45) XM(I),Vtc(I)
*       -Cpc output places suction surface in the 'down' position
          WRITE (81,45) XM(I),-Cpc
             CCPC(I) = CPC
      180     CONTINUE
      45     FORMAT (2(F10.5,2X))
      48     FORMAT (3(F10.5))
      49     FORMAT (3I5)
             WRITE (90,49) N,ISTAG,IANS
             DO 190 I=1,N
             WRITE (90,48) XM(I),YM(I),VtC(I)
      190    CONTINUE
             CALL FANDM(ALPHA,CL,CD,CM,CCPC,X,Y,N)
             WRITE (91,191)ALPHA*180./pi,CL,CD,CM
             write(6,1031)cl,cd,cm
      1031   format ( /,
          +     2x,'Cl = ',f10.6,/,
          +     2x,'Cd = ',f10.6,/,
          +     2x,'Cm = ',f10.6,/ )
      191    FORMAT(/,2X,F4.1,3(2X,F9.6))
             CLOSE(UNIT=88)
             CLOSE(UNIT=89)
             CLOSE(UNIT=81)
             CLOSE(UNIT=90)
             CLOSE(UNIT=91)
             print *,'CALCULATIONS COMPLETE'
             PRINT *,'OUTPUT FILES ARE vel.dat, cp.dat, CLDM.DAT, bl2d.dat'
             STOP
             END
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      C      SUBROUTINE FANDM
      C             INTEGRATE PRESSURE DISTRIBUTION BY TRAPEZOIDAL RULE
      CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      SUBROUTINE FANDM(ALPHA,cl,cd,cm,CCPC,x,y,n)
      REAL ALPHA,CL,CD,CM,CCPC(202),X(202),Y(202),CFX,CFY,XMID,YMID,
     :        DX,DY
      CFX     = 0.0
      CFY     = 0.0
      CM      = 0.0
      DO 100  I = 1,N
***    moment coeff is computed around pivot point at 25% Chord
      XMID    = .5*(X(I) + X(I+1)) - 0.25
      YMID    = .5*(Y(I) + Y(I+1))
      DX      = X(I+1) - X(I)
      DY      = Y(I+1) - Y(I)
      CFX     = CFX + CCPC(I)*DY
      CFY     = CFY - CCPC(I)*DX
      CM      = CM + CCPC(I)*(DX*XMID + DY*YMID)
 100  CONTINUE
      CD      = CFX*COS(ALPHA) + CFY*SIN(ALPHA)
      CL      = CFY*COS(ALPHA) - CFX*SIN(ALPHA)
      RETURN
      END

******************************************************************
* Gauss elimination procedure
     SUBROUTINE GAUSS(N,Z)
      INTEGER PV
      REAL Z(1:202,1:203),E
      E=1.0
10    IF (1.0+E.GT.1.0) THEN
      E=E/2.0
      GOTO 10
      END IF
      E=E*2
      EPS2 = 2 * E
c      PRINT *,'          MACHINE EPSILON=',E
1005  DET= 1
      DO 1010 I=1,N-1
      PV=I
      DO 1020 J=I+1,N
         IF (ABS(Z(PV,I))  .LT.  ABS(Z(J,I))) PV=J
1020     CONTINUE
      IF (PV.EQ.I) GOTO 1050
      DO 1040 JC=1,N+1
         TM=Z(I,JC)
         Z(I,JC)=Z(PV,JC)
         Z(PV,JC)=TM
1040     CONTINUE
1045     DET=-1*DET
1050     IF (Z(I,I).EQ.0) THEN
         GOTO 1200
      END IF
      DO 1060 JR=I+1, N
         IF (Z(JR,I).NE.0) THEN
            R=Z(JR,I)/Z(I,I)
            DO 1075 KC=I+1,N+1
            TEMP=Z(JR,KC)
            Z(JR,KC)=Z(JR,KC)-R*Z(I,KC)
            IF (ABS(Z(JR,KC)).LT.EPS2*TEMP) Z(JR,KC)=0.0

C                      !-- if the result of subtraction is smaller than
C                      !-- 2 times machine epsilon times the original
C                      !-- value, it is set to zero.
1075           CONTINUE
         END IF
1060     CONTINUE
1010  CONTINUE
      DO 1084 I=1,N
      DET=DET*Z(I,I)
1084  CONTINUE
      IF (Z(N,N).EQ.0) GOTO 1200
      Z(N,N+1)=Z(N,N+1)/Z(N,N)
      DO 1130 NV=N-1,1,-1
      VA=Z(NV,N+1)
      DO 1120 K=NV+1,N
         VA=VA-Z(NV,K)*Z(K,N+1)
1120     CONTINUE
      Z(NV,N+1)=VA/Z(NV,NV)
1130  CONTINUE
```

```
      RETURN
1200  PRINT *,'MATRIX IS SINGULAR'
      PRINT *,'I=',I,'Z(I,I)=',Z(I,I)
      STOP
      END


****************************************************************
****************************************************************
      PROGRAM BL2D
****************************************************************
*            CDR  T. Johnston   (July 1993)
****************************************************************
*     XCTRI(1)   =  Input for upper transition pt from panel
*     XCTRI(2)   =  Input for Lower transition pt from panel
*     NI         =   # Panels
*     IS         =  Stagnation Pt
*     ITRANS     =  Default/USER input for transiton flag
*     XI(I)      =  Control pts
*     YI(I)      =  Control pts
*     VEI(I)     =  Tangential Velocities
*     NXTSF(1)   =  # panels on upper surface
*     NXTSF(2)   =  # panels on lower surface
*     NXT        =  # panels on a surface in 200 ISF loop
*     ISF        =  Surface flag (1=upper, 2=Lower)
*     XC(I)      =  Redefined Control pts
*     YC(I)      =  Redefined Control pts
*     X(I)       =  Original nodes
***************************************************************
      COMMON /BLC0/ RL,TRANSNEW(2),NBL(2),XCTRI(2),ntflag,NI
      COMMON /BLC1/ XCTR,XC(200),YC(200),itr
      COMMON /BLC2/ NX,NXT,NP,NPT,NTR,IT,ISF
      COMMON /BLC3/ X(200),UE(200),P1(200),P2(200),GMTR(200)
      COMMON /BLCS/ DLS(200),VW(200),CF(200),THT(200)
      integer ntflag,ni,nx,np,nxt,is,itrans,isf,it,npt,ntr
      DIMENSION     NXTSF(2),XI(200),YI(200),VEI(200)
      CHARACTER  VAR*25
      REAL  PGAMTR
      OPEN (UNIT=9,FILE='bl2d.dat',STATUS='UNKNOWN')
      OPEN (UNIT=8,FILE='bl2d.out',STATUS='UNKNOWN')
      OPEN (UNIT=20,FILE='cf.dat',STATUS='UNKNOWN')
      OPEN (UNIT=21,FILE='dls.dat',STATUS='UNKNOWN')
      OPEN (UNIT=54,FILE='pro1.dat',STATUS='UNKNOWN')
      OPEN (UNIT=55,FILE='pro2.dat',STATUS='UNKNOWN')
**        READ BL2D.DAT
      WRITE(6,*) 'READING THE DATA...'
      READ ( 9,* ) RL,XCTRI(1),XCTRI(2)
      READ ( 9,* ) NI,IS,ITRANS
      READ ( 9,15 ) (XI(I),YI(I),VEI(I),I=1,NI)
      WRITE(6,*) 'INPUT OF DATA COMPLETE.'
***    Check Stagnation Point for adjustment
      print *
      print *,'    I Stagnation is '
      print *,IS
      print *
      print *,'    INPUT NEW I Stagnation = '
      READ *,IS
**     Write to Bl2d.out
 5    WRITE(8,90) RL,XCTRI(1),XCTRI(2)
      NXTSF(1)= NI - IS + 1
      NXTSF(2)= IS
***     DATA FOR EACH SURFACE (UPPER,LOWER)
      DO 200 ISF = 1,2
      ntflag=0
      NXT = NXTSF(ISF)
      GO TO (201,202),ISF
 ***   REDEFINE CONTROL POINTS
C     UPPER SURFACE
  201 II = IS-1
       DO 211 I=1,NXT
         II = II+1
         XC(I) = XI(II)
         YC(I) = YI(II)
         UE(I) = VEI(II)
```

```
  211 CONTINUE
      GO TO 300
C     LOWER SURFACE
  202 II = IS+1
      DO 212 I=1,NXT
         II = II-1
         XC(I) = XI(II)
         YC(I) = YI(II)
         UE(I) = VEI(II)
  212 CONTINUE
  300  X(1) = 0.0
***     DEFINE NODES
       DO 301 I=2,NXT
 301  X(I) = X(I-1)+SQRT((XC(I)-XC(I-1))**2+(YC(I)-YC(I-1))**2)
C     TRANSITION LOCATION
       DO 320 I=1,NXT
         GMTR(I) = 0.0
         IF (XC(I) .GE. XCTRI(ISF)) GO TO 321
  320 CONTINUE
***    NTR IS TRANSITION PANEL
   321 NTR = I
***    CEBECI-SMITH TURBULENCE MODEL
      PGAMTR  = 1200.
      RXNTR   = X(NTR-1)* UE(NTR-1) * RL
      GGFT    = RL**2/RXNTR**1.34*UE(NTR-1)**3
      UEINTG  = 0.0
      U1      = 0.5/UE(NTR-1)/ PGAMTR
      DO 322 I = NTR,NXT
         U2       = 0.5/UE(I)/PGAMTR
         UEINTG  = UEINTG+(U1+U2)*(X(I)-X(I-1))
         U1       = U2
         GG       = GGFT*UEINTG*(X(I)-X(NTR-1))
         IF(GG .GT. 10.0) GO TO 323
         GMTR(I)= 1.0-EXP(-GG)
  322 CONTINUE
  323 DO 324 II=I,NXT
  324 GMTR(II) = 1.0
C     PRESSURE GRADIENT PARAMETERS
      DX      = X(2)-X(1)
      DUE     = UE(2)-UE(1)
      ANG2    = ATAN2(DUE,DX)
      DL2     = DX
      DO 331 I = 2,NXT-1
         ANG1     = ANG2
         DL1      = DL2
         DX       = X(I+1)-X(I)
         DUE      = UE(I+1)-UE(I)
         ANG2     = ATAN2(DUE,DX)
         DL2      = DX
         ANG      = (DL2*ANG1+DL1*ANG2)/(DL1+DL2)
         P2(I)    = TAN(ANG)
  331 CONTINUE
      P2(NXT) = 2.*DUE/DL2 - P2(NXT-1)
      DO 330 I = 2,NXT
      P2(I)   = X(I) * P2(I) /UE(I)
      P1(I)   = 0.5 * (1.0 + P2(I))
  330 CONTINUE
      P2(1)   = 1.0
      P1(1)   = 0.5 * (1.0 + P2(1))
**    BOUNDARY LAYER CALCULATION
      WRITE(6,*) 'BOUNDARY LAYER COMPUTATIONS IN PROGRESS..'
      CALL BL
      WRITE(8,910)ISF,(I,XC(I),X(I),VW(I),CF(I),DLS(I),THT(I),I=1,NXT)
      if(ISF.EQ.1) then
      write(20,905) (XC(I),CF(I),I=2,NXT)
      write(21,905) (XC(I),DLS(I),I=2,NXT)
      end if
  905 FORMAT(F8.4,4X,E11.4)
  200 CONTINUE
***IF AOA is 0 deg., make trans. locs. equal:
      if(vei(2).eq.vei(ni-1)) transnew(1)=transnew(2)
***  Display Michels transition estimate
c      if(ITRANS.eq.0) then
         print *, 'MICHELS TRANSITION ESTIMATE ****'
         print *,'Estimate for upper transition:',transnew(1)
         print *,'Estimate for lower transition:',transnew(2)
c      END IF
      CLOSE(UNIT=8)
```

```
      CLOSE(UNIT=9)
      CLOSE(UNIT=20)
      CLOSE(UNIT=21)
      CLOSE(UNIT=54)
      CLOSE(UNIT=55)
 10   FORMAT(3I5)
 15   FORMAT(3F10.0)
 90   FORMAT(//5X,'RL=',E12.5,5X,'XCTRI(1) =',F8.3,5X,'XCTR(2) =',F8.3)
 95   FORMAT(5X,A25,2X,f8.7)
 910  FORMAT(///2X,'*** SUMMARY OF BOUNDARY LAYER SOLUTIONS OF ISF =',I2
     +//2X,'NX',4X,'XC',8X,'S',8X,'VW',8X,'CF',8X,'DLS',8X,'THT'
     +/(I5,2F8.4,4E11.4))
      print*
      print*,'Output files: cf.dat, dls.dat, pro1.dat,pro2.dat,bl2d.dat'
      STOP
      END
************************************************************************

      SUBROUTINE BL
      COMMON /BLC2/ NX,NXT,NP,NPT,NTR,IT,ISF
      COMMON /BLC3/ X(200),UE(200),P1(200),P2(200),GMTR(200)
      COMMON /BLC7/ ETA(201),DETA(201),A(201)
      COMMON /BLC8/ F(201,2),U(201,2),V(201,2),B(201,2)
      COMMON /BLC6/ DELF(201),DELU(201),DELV(201)
      integer ntflag,ni,nx,np,nxt,is,itrans,isf,it,npt,ntr
      NX     = 0
      ITMAX  = 10
      IGROWT = 2
      EPSL   = 0.0001
      EPST   = 0.01
      NPT    = 101
C     ETA-GRID
      ETAE    = 8.0
      VGP     = 1.10
      DETA(1) = 0.01
      NP = LOG((ETAE/DETA(1))*(VGP-1.0)+1.0)/LOG(VGP)+1.001
      ETA(1) = 0.0
      DO 10 J=2,NPT
         ETA(J) = ETA(J-1) + DETA(J-1)
         DETA(J)= VGP*DETA(J-1)
         A(J)   = 0.5*DETA(J-1)
 10   CONTINUE
C     INITIAL LAMINAR VELOCITY PROFILE
      DO 20  J=1,NP
         ETAB = ETA(J)/ETA(NP)
         ETAB2 = ETAB**2
         F(J,2) = 0.25*ETA(NP)*ETAB2*(3.0 - 0.5*ETAB2)
         U(J,2) = 0.5*ETAB*(3.0 - ETAB2)
         V(J,2) = 1.5*(1.0 - ETAB2)/ETA(NP)
         B(J,2) = 1.0
 20   CONTINUE
**  Step thru all panels (NX to NXT)
  1   NX      = NX+1
      IT      =0
      IGROW   = 0
  5   IT      = IT+1
      IF (IT .GT. ITMAX) GO TO 101
      IF (NX .GE. NTR) CALL EDDY
      CALL COEF
      CALL SOLV3
C     CHECK FOR CONVERGENCE
      IF (NX .LT. NTR) THEN
         IF(ABS(DELV(1)) .GT. EPSL) GO TO 5
      ELSE
         IF(ABS(DELV(1)/V(1,2)) .GT. EPST) GO TO 5
      ENDIF
C     PROFILES FOR GROWTH
 99   DO 30 J=NP+1,NPT
         F(J,2) = F(J-1,2) + DETA(J-1)*U(J-1,2)
         U(J,2) = U(J-1,2)
         V(J,2) = 0.0
         B(J,2) = B(J-1,2)
 30   CONTINUE
C     CHECK FOR GROWTH
      IF (ABS(V(NP,2)) .GT. 0.0005 .OR. ABS(1.0-U(NP-2,2)/U(NP,2))
     +              .GT. 0.005) THEN
         NP = NP+2
         IGROW  = IGROW+1
```

```
          IF (NP .LE. NPT .AND. IGROW .LE. IGROWT) THEN
             IT     = 0
             GO TO 5
          ENDIF
       ENDIF
 101   CALL OUTPUT
       IF (NX .LT. NXT) GO TO 1
       RETURN
       END

*************************************************************

       SUBROUTINE COEF
       COMMON /BLC2/ NX,NXT,NP,NPT,NTR,IT,ISF
       COMMON /BLC3/ X(200),UE(200),P1(200),P2(200),GMTR(200)
       COMMON /BLC7/ ETA(201),DETA(201),A(201)
       COMMON /BLC8/ F(201,2),U(201,2),V(201,2),B(201,2)
       COMMON /BLC9/ S1(201),S2(201),S3(201),S4(201),S5(201),S6(201),
     +               S7(201),S8(201),R1(201),R2(201),R3(201),R4(201)
        integer ntflag,ni,nx,np,nxt,is,itrans,isf,it,npt,ntr
       P1H = 0.5 * P1(NX)
       IF (NX .EQ. 1) THEN
          CEL = 0.0
          CELH= 0.0
          DO 5 J=1,NP
             F(J,1) = 0.0
             U(J,1) = 0.0
             V(J,1) = 0.0
             B(J,1) = 0.0
   5      CONTINUE
       ELSE
          CEL = 0.5 * (X(NX)+X(NX-1))/(X(NX)-X(NX-1))
          CELH= 0.5 * CEL
       ENDIF
       DO 100 J= 2,NP
C         CURRENT STATION
          FB      = 0.5*(F(J,2) + F(J-1,2))
          UB      = 0.5*(U(J,2) + U(J-1,2))
          FVB     = 0.5*(F(J,2)*V(J,2)+F(J-1,2)*V(J-1,2))
          VB      = 0.5*(V(J,2) + V(J-1,2))
          USB     = 0.5*(U(J,2)**2 + U(J-1,2)**2)
          DERBV   =(B(J,2)*V(J,2) - B(J-1,2)*V(J-1,2))/DETA(J-1)
C         PREVIOUS STATION
          CFB     = 0.5*(F(J,1) + F(J-1,1))
          CUB     = 0.5*(U(J,1) + U(J-1,1))
          CVB     = 0.5*(V(J,1) + V(J-1,1))
          CUSB    = 0.5*(U(J,1)**2 + U(J-1,1)**2)
          CFVB    = 0.5*(F(J,1)*V(J,1)+F(J-1,1)*V(J-1,1))
          CDERBV = (B(J,1)*V(J,1) - B(J-1,1)*V(J-1,1))/DETA(J-1)
C         S- COEFFICIENTS
          S1(J)   = CELH*(F(J,2) - CFB) + P1H*F(J,2) + B(J,2)/DETA(J-1)
          S2(J)   = CELH*(F(J-1,2)-CFB) + P1H*F(J-1,2)-B(J-1,2)/DETA(J-1)
          S3(J)   = CELH*(V(J,2) + CVB) + P1H*V(J,2)
          S4(J)   = CELH*(V(J-1,2) + CVB) + P1H*V(J-1,2)
          S5(J)   = -(CEL+P2(NX))*U(J,2)
          S6(J)   = -(CEL+P2(NX))*U(J-1,2)
C         R- COEFFICIENTS
          IF (NX .EQ. 1) THEN
             CRB     = -P2(NX)
             R2(J)   = CRB - (DERBV + P1(NX)*FVB - P2(NX)*USB)
          ELSE
             CLB     = CDERBV  + P1(NX-1)*CFVB - P2(NX-1)*CUSB + P2(NX-1)
             CRB     = -CLB - CEL*CUSB - P2(NX)
             R2(J)   = CRB - (DERBV + P1(NX)*FVB- (CEL+P2(NX))*USB  + CEL*
     +                 (FVB + CVB*FB - VB*CFB - CFVB))
          ENDIF
          R1(J)   = F(J-1,2) - F(J,2) + DETA(J-1)*UB
          R3(J-1)= U(J-1,2) - U(J,2) + DETA(J-1)*VB
 100   CONTINUE
C      BOUNDARY CONDITIONS
       R1(1)   = 0.0
       R2(1)   = 0.0
       R3(NP) = 0.0
       RETURN
       END

*************************************************************
*****************************************************************
       SUBROUTINE EDDY
```

```
      COMMON /BLC0/ RL,transnew(2),NBL(2),XCTRI(2),ntflag,NI
      COMMON /BLC2/ NX,NXT,NP,NPT,NTR,IT,ISF
      COMMON /BLC3/ X(200),UE(200),P1(200),P2(200),GMTR(200)
      COMMON /BLC7/ ETA(201),DETA(201),A(201)
      COMMON /BLC8/ F(201,2),U(201,2),V(201,2),B(201,2)
       integer ntflag,ni,nx,np,nxt,is,itrans,isf,it,npt,ntr
      DIMENSION EDVI(201)
      RL2    = SQRT(RL*UE(NX)*X(NX))
      RL4    = SQRT(RL2)
      RL216  = 0.16 * RL2
      ALFA   = 0.0168
      EDVO   = ALFA*RL2*GMTR(NX)*(U(NP,2)*ETA(NP)-F(NP,2))
      EDVI(1)= 0.0
      YBAJ   = RL4*SQRT(ABS(V(1,2)))/26.0
      DO 70 J=2,NP
         JJ      = J
         YBA     = YBAJ*ETA(J)
         EL      = 1.0
         IF(YBA .LT. 10.0) EL = 1.0 - EXP(-YBA)
         EDVI(J) = RL216*GMTR(NX)*(EL*ETA(J))**2 * ABS(V(J,2))
         IF(EDVI(J) .GT. EDVO) GO TO 90
         IF (EDVI(J) .LE. EDVI(J-1)) EDVI(J)= EDVI(J-1)
         B(J,2) = 1.0 + EDVI(J)
   70 CONTINUE
   90 DO 100 JJ=J,NPT
  100 B(JJ,2) = 1.0 + EDVO
      B(1,2) = 1.0
      RETURN
      END

*************************************************************
      SUBROUTINE OUTPUT
      COMMON /BLC0/ RL,transnew(2),NBL(2),XCTRI(2),ntflag,NI
      COMMON /BLC2/ NX,NXT,NP,NPT,NTR,IT,ISF
      COMMON /BLC3/ X(200).UE(200),P1(200),P2(200),GMTR(200)
      COMMON /BLC7/ ETA(201),DETA(201),A(201)
      COMMON /BLC8/ F(201,2),U(201,2),V(201,2),B(201,2)
      COMMON /BLCS/ DLS(200),VW(200),CF(200),THT(200)
      integer ni,nx,np,nxt,is,itrans,isf,it,npt,ntr,ntflag,nstop
      dimension rdiff(201),rdlow(201)
      IF(NX.EQ.1 ) THEN
         DLS(NX)= 0.0
         THT(NX)= 0.0
         CF(NX) = 0.0
         VW(NX) = V(1,2)
         rdifflow=1000
         nstop=0
      ELSE
         SQRX    = SQRT(UE(NX)*X(NX)*RL)
         CF(NX) = 2.0 * V(1,2) * B(1,2) /SQRX
         VW(NX) = V(1,2)
         DLS(NX)= X(NX)/SQRX * (ETA(NP)-F(NP,2))

         U1      = U(1,2) * (1.0 -U(1,2))
         SUM     = 0.0
         DO 20 J=2,NP
            U2      = U(J,2) * (1.0 -U(J,2))
            SUM     = SUM + A(J) * (U1 + U2)
            U1      = U2
20       CONTINUE
         THT(NX)= X(NX)/SQRX * SUM
         rex=UE(NX)*X(NX)*RL
         rtheta=UE(NX)*THT(NX)*RL
***  Michels's Criterian (JUST CALCULATED **NEVER** USED)
         rtrans=1.174*(1.0+22400.0/rex)*rex**0.46
         rdiff(nx)=abs(rtheta-rtrans)
         if ((NX.gt.2) .and. (rdlow(nx-1).eq.rdlow(nx-2))) then
             if (rdlow(nx-2).eq.rdlow(nx-3)) nstop=1
         endif
         if (ISF.eq.2) then
           if ((ntflag.eq.1) .and. (nstop.eq.0)) then
               transnew(ISF)=rex/(RL*UE(NX))
               ntflag=0
           endif
         endif
         if((rdiff(nx).LT.rdifflow) .and. (nstop.eq.0)) then
              transnew(ISF)=rex/(RL*UE(NX))
              rdifflow=rdiff(nx)
```

```
                 ntflag=1
              endif
              rdlow(nx)=rdifflow
          ENDIF
C         SHIFT PROFILES FOR THE NEXT STATION
          ymark=.0005
          DO 175 J=1,NPT
             if(ISF.EQ.1) then
                if(U(J,1).LT.(0.995)) then
                   lasty=1
                   yplot=ETA(J)*SQRT(X(NX)/(RL*UE(NX)))
                      do nxloop=5,NI/2-1,5
                       if(NX.EQ.nxloop) then
 91                       markx=NX/5
                          numw=markx+30
c                         write (numw,*) U(J,1)+markx,yplot
                    write (54,*) U(J,1)+markx,yplot
                    if(yplot.gt.ymark) then
                     write(55,*) markx,ymark
                            ydiff=yplot-yplotold
                            udiff=U(J,1)-U(J-1,1)
                            xvalue=U(J-1,1)+udiff*(ymark-yplotold)/ydiff
                    write(55,*) xvalue+markx,ymark
                    write(55,92)
 92                         format (/)
                            ymark=ymark+.0005
                            if(yplot.GT.ymark) goto 91
                         endif
                      endif
                   end do
                else
                   if (lasty.EQ.1) then
                      lasty=0
                      do m=1,2
                         do nxloop=5,NI/2-1,5
                            if(NX.EQ.nxloop) then
                               markx=NX/5
                               numw=markx+30
c                              write (numw,*) markx,yplot
                    write (54,*) markx,yplot
                            endif
                         end do
                         yplot=0.0
                      end do
                   endif
                endif
             endif
             yplotold=yplot
             F(J,1)  = F(J,2)
             U(J,1)  = U(J,2)
             V(J,1)  = V(J,2)
             B(J,1)  = B(J,2)
 175      CONTINUE
          RETURN
          END

**********************************************************************
          SUBROUTINE SOLV3
          COMMON /BLC2/ NX,NXT,NP,NPT,NTR,IT,ISF
          COMMON /BLC7/ ETA(201),DETA(201),A(201)
          COMMON /BLC8/ F(201,2),U(201,2),V(201,2),B(201,2)
          COMMON /BLC9/ S1(201),S2(201),S3(201),S4(201),S5(201),S6(201),
         +              S7(201),S8(201),R1(201),R2(201),R3(201),R4(201)
          COMMON /BLC6/ DELF(201),DELU(201),DELV(201)
           integer ntflag,ni,nx,np,nxt,is,itrans,isf,it,npt,ntr
          DIMENSION     A11(201),A12(201),A13(201),A14(201),
         +              A21(201),A22(201),A23(201),A24(201)
          A11(1)= 1.0
          A12(1)= 0.0
          A13(1)= 0.0
          A21(1)= 0.0
          A22(1)= 1.0
          A23(1)= 0.0
          G11   =-1.0
          G12   =-A(2)
          G13   = 0.0
          G21   = S4(2)
          G23   =-S2(2)/A(2)
```

```
          G22    = G23+S6(2)
          A11(2)= 1.0
          A12(2)=-A(2)-G13
          A13(2)= A(2)*G13
          A21(2)= S3(2)
          A22(2)= S5(2)-G23
          A23(2)= S1(2)+A(2)*G23
          R1(2)  = R1(2)-(G11*R1(1)+G12*R2(1)+G13*R3(1))
          R2(2)  = R2(2)-(G21*R1(1)+G22*R2(1)+G23*R3(1))
C  FORWARD SWEEP
            DO 500 J=2,NP
C           IF(DEN .GT. 1E12) GO TO 11
           DEN    = (A13(J-1)*A21(J-1)-A23(J-1)*A11(J-1)-A(J)*
     +                 (A12(J-1)*A21(J-1)-A22(J-1)*A11(J-1)))
C 11        IF(DEN1 .GT. 1E12) GO TO 12
           DEN1   = A22(J-1)*A(J)-A23(J-1)
C 12        PRINT*, 'DEN,DEN1=',DEN,DEN1
           G11    = (A23(J-1)+A(J)*(A(J)*A21(J-1)-A22(J-1)))/DEN
           G12    = -(A(J)*A(J)+G11*(A12(J-1)*A(J)-A13(J-1)))/DEN1
           G13    = (G11*A13(J-1)+G12*A23(J-1))/A(J)
           G21    = (S2(J)*A21(J-1)-S4(J)*A23(J-1)+A(J)*(S4(J)*
     +                 A22(J-1)-S6(J)*A21(J-1)))/DEN
           G22    = (-S2(J)+S6(J)*A(J)-G21*(A(J)*A12(J-1)-A13(J-1)))/DEN1
           G23    = G21*A12(J-1)+G22*A22(J-1)-S6(J)
           A11(J)= 1.0
           A12(J)=-A(J)-G13
           A13(J)= A(J)*G13
           A21(J)= S3(J)
           A22(J)= S5(J)-G23
           A23(J)= S1(J)+A(J)*G23
           R1(J)  = R1(J)-(G11*R1(J-1)+G12*R2(J-1)+G13*R3(J-1))
           R2(J)  = R2(J)-(G21*R1(J-1)+G22*R2(J-1)+G23*R3(J-1))
           IF(R2(J) .GT. 1E20) THEN
             RCHK = R2(J)
             GO TO 99
           END IF
500    CONTINUE
C  BACKWARD SWEEP
       DELU(NP) = R3(NP)
       E1        = R1(NP)-A12(NP)*DELU(NP)
       E2        = R2(NP)-A22(NP)*DELU(NP)
       DELV(NP)  = (E2*A11(NP)-E1*A21(NP))/(A23(NP)*A11(NP)-A13(NP)*
     +                 A21(NP))
       DELF(NP)  = (E1-A13(NP)*DELV(NP))/A11(NP)
       DO 600 J = NP-1,1,-1
          E3         = R3(J)-DELU(J+1)+A(J+1)*DELV(J+1)
          DEN2       = A21(J)*A12(J)*A(J+1)-A21(J)*A13(J)-A(J+1)*A22(J)*
     +               A11(J)+A23(J)*A11(J)
          DELV(J)    = (A11(J)*(R2(J)+E3*A22(J))-A21(J)*R1(J)-E3*A21(J)*
     +               A12(J))/DEN2
          DELU(J)    =-A(J+1)*DELV(J)-E3
          DELF(J)    = (R1(J)-A12(J)*DELU(J)-A13(J)*DELV(J))/A11(J)
600    CONTINUE
99     DO 700 J=1,NP
          F(J,2)= F(J,2)+DELF(J)
          U(J,2)= U(J,2)+DELU(J)
          V(J,2)= V(J,2)+DELV(J)
700    CONTINUE
       U(1,2)= 0.0

       RETURN
       END
```

APPENDIX B

A. UPOT.IN NAME LIST

```
3
***************************************************************
*          AIRFOIL TYPE : NACA 0012   (RAMP)
***************************************************************
IFLAG  NLOWER  NUPPER
 0       50      50
 12
IRAMP  IOSCIL   ALPI     ALPMAX      RFREQ      PIVOT
 1      0       0.0     30.0        0.01       0.25
IGUST  UGUST  VGUST
 0      0.     0.
ITRANS  DELHX  DELHY  PHASE
 0       0.00   0.00   0.00
CYCLE  NTCYCLE     TOL
 1.5     125     0.0001
naot &  naot X aoa values multiplied by 10 (integer)
 8      1,60,90,110,130,150,170,200
***************************************************************
Comments...

IFLAG   0:  NACA 4 or 5 digit airfoil (program computes coordinates)
        1:  User suplies surface coordinates
                NLOWER:  # panels upper surface
                NUPPER:  # panels lower surface
                        (NOTE:  Next line entry is either a NACA 4,5 digit
                                airfoil or user supplied coordinates (no blanks)

IRAMP  0: n/a               *** RFREQ is based on full chord
       2: Straight ramp     *** RFREQ = A = Reduced pitch rate
       1: Modified ramp     ***             = αc/U

IOSCIL 0: n/a               *** RFREQ is based on full chord
       1: Sinusoidal pitch, motion starts at min Aoa
                           ***  RFEQ = k = Reduced Frequency
                                         = ωc/U
ITRANS 0: n/a
       1: Translational harmonic oscillatio

CYCLE  : # of cycles for oscillatory motions
         in case of ramp, cycle=1.5 denotes airfoil is held
         at max aoa for the duration of .5 cycle

NTCYCLE: # of time steps for each cycle
          CYCLE*NTCYCLE is limited to 200 currently.

NAOT: # of input aoa for cp output
       - angles should be increasing order,
       - for oscilatory motions angles should increase
         first then decrease, decreasing angles are for
         return cycle..
```

```
3
***************************************************************
         AIRFOIL TYPE : SSC-A09   (SINUSOID)
***************************************************************
  IFLAG  NLOWER  NUPPER
   1       92      92
 1.000000  .0004815
 .975084  -.001325
 .955144  -.002610
 .935204  -.004290
 .915264  -.006081
    *         *
    *         *
    *         *
 .915264   .009046
 .935204   .006229
```

```
 .955144    .003849
 .975084    .002288
 1.00000    .0004815
IRAMP  IOSCIL   ALPI     ALPMAX       RFREQ       PIVOT
 0        1       0        20         0.1        0.25
IGUST  UGUST  VGUST
 0      0.     0.
ITRANS  DELHX  DELHY  PHASE
 0       0.00   0.00    0.00
CYCLE  NTCYCLE      TOL
 1.0     160      0.0001
naot &  naot X aoa values multiplied by 10 (integer)
15   1,60,90,110,130,150,170,200,170,150,130,110,90,60,1
**************************************************************
Comments...

IFLAG   0:  NACA 4 or 5 digit airfoil (program computes coordinates)
        1:  User suplies surface coordinates
                 NLOWER:  # panels upper surface
                 NUPPER:  # panels lower surface
                           (NOTE:  Next line entry is either a NACA 4,5 digit
                                   airfoil or user supplied coordinates (no blanks)

IRAMP  0: n/a               *** RFREQ is based on full chord
       2: Straight ramp     *** RFREQ = A = Reduced pitch rate
       1: Modified ramp     ***            = αc/U

IOSCIL 0: n/a               *** RFREQ is based on full chord
       1: Sinusoidal pitch, motion starts at min Aoa
                           ***   RFEQ = k = Reduced Frequency
                                          = ωc/U
ITRANS 0: n/a
       1: Translational harmonic oscillatio

CYCLE  : # of cycles for oscillatory motions
         in case of ramp, cycle=1.5 denotes airfoil is held
         at max aoa for the duration of .5 cycle

NTCYCLE: # of time steps for each cycle
          CYCLE*NTCYCLE is limited to 200 currently.

NAOT: # of input aoa for cp output
        - angles should be increasing order,
        - for oscilatory motions angles should increase
          first then decrease, decreasing angles are for
          return cycle..
```

B. PROGRAM UPOT.F SOURCE CODE

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C            CDR  T. Johnston
C                August 1993
C         PROGRAM UPOT (U2DIIF2, version 2.b, June 87)
c         modified by Tuncer     feb-1993
C
C         UNSTEADY MOTION OF A TWO-DIMENSIONAL AIRFOIL
C         IN INCOMPRESSIBLE INVISCID FLOW
C         USING PANEL METHODS BASED ON THE HESS & SMITH
C
C         THIS VERSION INCORPORATES THE FIRST MODIFICATION
C         TO THE ORIGINAL U2DIIF PROGRAM IN ITS COMPUTATION SPEEDS
C         CHANGES INCLUDE :
C         (A)  REMOVING PARTIAL PIVOTING IN GAUSSIAN ELIMINATION
C         (B)  OTHER THAN THE VERY FISRT COMPUTATION TIME-STEP,
C              ALL ELIMINATION PROCESSES ARE DONE ON THE R.H.S. ONLY
C         (C)  SIX 201*200 ARRAYS HAVE BEEN SAVED
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      parameter (nwmx=200, npmx=201)
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(npmx+1),Y(npmx+1),
     +             COSTHE(npmx),SINTHE(npmx),SS,NP1,NP2
      COMMON /WAK/ VYW,VXW,WAKE,DT
      COMMON /WAK2/ VYWK,VXWK
      COMMON /SING/ Q(nwmx),GAMMA,QK(nwmx),GAMK
      COMMON /CORV/ CV(nwmx),XC(nwmx),YC(nwmx),nt,TD,
     >              CVVX(nwmx),CVVY(nwmx)
```

```
      COMMON /POT/ PHI(nwmx),PHIK(nwmx)
      COMMON /GUST/ UG(nwmx),VG(nwmx),XGF,UGUST,VGUST
      COMMON /EXTV/ UE(nwmx)
      COMMON /MAINout/ ialfao(20), naot, nao
      COMMON /CPD/ CP(200),pivot
      common /phase/  ta(nwmx),alphaa(nwmx),cla(nwmx),cda(nwmx),
     >                cma(nwmx),hya(nwmx)
      DIMENSION XXC(nwmx),YYC(nwmx)
      PI      = 3.1415926585
      open (unit=1,file='u.in',form='formatted')
      open (unit=8,file='cl.d',form='formatted')
      open (unit=9,file='cm.d',form='formatted')
      open (unit=91,file='v.d',form='formatted')
      open (unit=92,file='uout.d',form='formatted')
C..INPUT FROM FILE CODE 5 AND SET UP PANEL NODES AND SLOPES
      CALL INDATA
      CALL SETUP
      read(1,*)
      READ(1,*) IRAMP, IOSCIL, ALPI,ALPMAX,FREQ,PIVOT
      read(1,*)
      READ(1,*) IGUST, UGUST, VGUST
      read(1,*)
      READ(1,*) ITRANS,DELHX,DELHY,PHASE
      read(1,*)
      READ(1,*) cycle,ntcycle,TOL
      read(1,*)
      READ(1,*) naot, (ialfao(i), i=1,naot)
      nao = 1
      if(iramp .gt. 0 .or. ioscil .gt. 0) then
        if(iramp .gt. 0) print*,' RAMP MOTION, IRAMP      = ',iramp
        if(ioscil .gt. 0) print*,
     >                  ' OSCILLATORY MOTION, IOSCIL    = ',ioscil
        WRITE (6,555) ALPI,alpmax,FREQ,PIVOT
  555 FORMAT (2X, 'INITIAL ANGLE OF ATTACK         =',F10.4,/,
     >        2X, 'FINAL   ANGLE OF ATTACK         =',F10.4,/,
     >        2X, 'REDUCED FREQ. FOR OSCIL         =',F10.4,/,
     >        2X, 'PIVOT POINT                     =',F10.4,/,
     >        2X, '=======================================')
        dalp = alpmax-alpi
**         tcon =  2(pi)/k
**            k =  (w c) / 2 Vinf  =  (w A) / alfadot
**      alfadot =       w          =  (2 u A) / c
        tcon = 2.*pi/freq
      endif
      if(igust .eq. 1 ) then
         WRITE (6,558) ugust, vgust
  558 FORMAT (2X, 'STREAMWISE GUST VELOCITY       =',F10.4,/,
     >        2X, 'PERPENDICULAR GUST VELOCITY    =',F10.4,/,
     >        2X, '=======================================')
         ANGLE    = ALPI*PI/180. + ATAN(VGUST/(1.+UGUST))
         COSAng   = COS(ANGLE)
         SINAng   = SIN(ANGLE)
      endif
      if(itrans .eq. 1 )  then
         WRITE (6,556) DELHX,DELHY,PHASE
  556 FORMAT (2X, 'X AMPLITUDE OF TRANS OSCILL. =',F10.4,/,
     >        2X, 'Y AMPLITUDE OF TRANS OSCILL. =',F10.4,/,
     >        2X, 'PHASE       OF TRANS OSCILL. =',F10.4,/,
     >        2X, '=======================================')
         tcon = 2.*pi/freq
      endif
      WRITE (6,557) cycle,ntcycle,TOL
  557 FORMAT (2X, 'TOTAL # OF CYCLES               =',F10.4,/,
     >        2X, '# of TIME STEPS PER CYCLE       =',i6   ,/,
     >        2X, 'TOLERANCE FOR CONVERGENCE       =',F10.4,/,
     >        2X, '=======================================')
       dts  = tcon/float(ntcycle)
**  float()   >>    Numeric to real
C.STEADY FLOW CALCULATION AT ALPI
      ALPHA    = ALPI
      WRITE (6,1030) ALPHA
 1030 FORMAT (//,' STEADY FLOW SOLUTION AT ALPHA = ',F10.6,/)
      WRITE (6,1032)
 1032 format (//,4x,'I    XMID      Q(I)      GAMMA    CP(I)    UE(I) ',/)
     IF (ALPHA .GT. 90.) stop   "ALpha .gt. 90 degres"
      COSALF  = COS(ALPHA*PI/180.)
      SINALF  = SIN(ALPHA*PI/180.)
      CALL COFISH(SINALF,COSALF)
```

```
          CALL INFL (0)
          CALL GAUSS(1,0,0)
          CALL VELDIS(SINALF,COSALF)
          CALL PINTEG (GAMMA,Q)
          CALL PRESSs
          CALL FANDM(SINALF,COSALF,cl,cd,cm)
          write(6,1031)cl,cd,cm
 1031     format ( /,
     +        5x,'Cl = ',f10.6,/,
     +        5x,'Cd = ',f10.6,/,
     +        5x,'Cm = ',f10.6,/ )
          if(cycle .le. 0. ) stop   " Steady solution only"
C..INITIALIZATION FOR UNSTEADY FLOW CALCULATION
          HX       = 0.0
          HY       = 0.0
          HXO      = 0.0
          HYO      = 0.0
          DHX      = 0.0
          DHY      = 0.0
          UX       = 0.0
          UY       = 0.0
          ALP = ALPI
          DA       = 0.0
          COSDA    = 1.0
          SINDA    = 0.0
          OMEGA    = 0.0
          XGF      = 0.0
          PHA      = PHASE*PI/180.
          VXW      = COSALF
          VYW      = SINALF
          GAMK     = GAMMA
          T        = 0.0
          TOLD     = 0.0
          DT       = DTS
          TD       = DTS
          WRITE (6,1051)
 1051     FORMAT (/////,' ****************************************',/,
     +    ' ***  BEGIN UNSTEADY FLOW SOLUTION  ****',/,
     +    ' ****************************************'//
     >    ' istep  alpha     time      nitr       cl         cd         cm'/)
          ntmax = min(nwmx, int(cycle*ntcycle) )
**        Tmax for Ramp motion
**            Tmax = tcon/ntcycle*(ntmax-1.)
          DO NT = 1,ntmax
          ta(nt) = t
C.. STORE CORE VORTEX COORDINATES FOR TIME STEP ADJUSTMENTS
          if (nt .ne. 1) then
          DO 51   I = 1,nt-1
          XXC(I)  = XC(I)
   51     YYC(I)  = YC(I)
          endif
          IF (IRAMP .eq. 1) then
C. modified ramp change in aoa
             if (t .le. tcon) then
              DAL       = DALP * (3.-2.*T/TCON)*(T/TCON)**2
              ALPHA     = ALPI + DAL
          COSALF   = COS(ALPHA*PI/180.)
          SINALF   = SIN(ALPHA*PI/180.)
          DA       = ALPHA - ALP
          COSDA    = COS(DA*PI/180.)
          SINDA    = SIN(DA*PI/180.)
             OMEGA    = - (DALP*PI/180.) * (6.*T/(TCON*TCON)) * (1.-T/TCON)
          DHX      = PIVOT * (1.-COSDA)
          DHY      = - PIVOT * SINDA
          UY       = PIVOT * OMEGA
             else
          DAL      = 0.0
          ALPHA    = ALPmax
          COSALF   = COS(ALPHA*PI/180.)
          SINALF   = SIN(ALPHA*PI/180.)
          DA       = 0.0
          COSDA    = 1.0
          SINDA    = 0.0
          OMEGA    = 0.0
          DHX      = 0.0
          DHY      = 0.0
          UY       = 0.0
             endif
```

```
          ELSEIF (IRAMP .eq. 2) then
      C.straight ramp change in aoa
            if (t .le. tcon) then
          alpha    = alpi + dalp/tcon*t
          COSALF   = COS(ALPHA*PI/180.)
          SINALF   = SIN(ALPHA*PI/180.)
          DA       = ALPHA - ALP
          COSDA    = COS(DA*PI/180.)
          SINDA    = SIN(DA*PI/180.)
          OMEGA    = - dalp/tcon*(pi/180.)
          DHX      = PIVOT * (1.-COSDA)
          DHY      = - PIVOT * SINDA
          UY       = PIVOT * OMEGA
            else
          DAL      = 0.0
          ALPHA    = ALPmax
          COSALF   = COS(ALPHA*PI/180.)
          SINALF   = SIN(ALPHA*PI/180.)
          DA       = 0.0
          COSDA    = 1.0
          SINDA    = 0.0
          OMEGA    = 0.0
          DHX      = 0.0
          DHY      = 0.0
          UY       = 0.0
            endif
          ELSEIF (Ioscil .eq. 1) then
      C..rotational harmonic oscillation
      c      DAL      = DALP*SIN(FREQ*T)
      c      OMEGA    = - (DALP*PI/180.) * FREQ * COS(FREQ*T)
      c      ALPHA    = ALPI + DAL
             alpha    = alpi + 0.5*dalp*(1.- cos(freq*t) )
             omega    =  - 0.5*(dalp*pi/180.)*freq*sin(freq*t)
          COSALF   = COS(ALPHA*PI/180.)
          SINALF   = SIN(ALPHA*PI/180.)
          DA       = ALPHA - ALP
          COSDA    = COS(DA*PI/180.)
          SINDA    = SIN(DA*PI/180.)
          UY       = PIVOT * OMEGA
          DHX      = PIVOT * (1.-COSDA)
          DHY      = - PIVOT * SINDA
          ELSEIF (Igust .eq. 1) then
      C..sharp edge gust (ugust and/or vgust)
          XGF      = T
          DO 110  IG = 1,NODTOT
          UG(IG)   = 0.0
          VG(IG)   = 0.0
          XG       = X(IG)*COSALF + Y(IG)*SINALF
          XGP1     = X(IG+1)*COSALF + Y(IG+1)*SINALF
          IF (IG .LT. NLOWER+1) GO TO 120
          IF (XGF .LE. XG) GO TO 110
          IF (XGF .GE. XGP1) GO TO 111
          FAC      = (XGF - XG)/(XGP1 - XG)
          UG(IG)   = UGUST*FAC
          VG(IG)   = VGUST*FAC
          GO TO 110
        111    UG(IG)   = UGUST
          VG(IG)   = VGUST
          GO TO 110
        120    IF (XGF .LE. XGP1) GO TO 110
          IF (XGF .GE. XG) GO TO 121
          FAC      = (XGF - XGP1)/(XG - XGP1)
          UG(IG)   = UGUST*FAC
          VG(IG)   = VGUST*FAC
          GO TO 110
        121    UG(IG)   = UGUST
          VG(IG)   = VGUST
        110    CONTINUE
          IF (XGF .LE. COSALF) MGUST = nt
          ENDIF
          alphaa(nt) = alpha
          if (Itrans .eq. 1) then
      C..translation harmonic oscillation
          HX       = DELHX * SIN(FREQ*T + PHA)
          HY       = DELHY * SIN(FREQ*T)
      c      HX        =-DELHX * COS(FREQ*T + PHA)
      c   HY       =-DELHY * COS(FREQ*T)
          DHX      = HX - HXO
```

```
          DHY      = HY - HYO
          UX       = DELHX*FREQ*COS(FREQ*T+PHA)
          UY       = DELHY*FREQ*COS(FREQ*T)
c            UX       = DELHX*FREQ*SIN(FREQ*T+PHA)
c            UY       = DELHY*FREQ*SIN(FREQ*T)
            hya(nt) = hy
           endif
C..TRANSFORM CORE VORTEX COORDINATES W. R. T. NEW AIRFOIL POSITION
          IF (nt .ne. 1) then
          DO 90   I = 1,nt-1
          XC(I)    = XXC(I) + CVVX(I) * DT
          YC(I)    = YYC(I) + CVVY(I) * DT
          XCO      = XC(I)
          YCO      = YC(I)
          XC(I)    = XCO*COSDA - YCO*SINDA + DHX
     90     YC(I)    = XCO*SINDA + YCO*COSDA + DHY
          endif
C..CALCULATE THE TRAILING EDGE WAKE ELEMENT
          NITR     = 0
   10     WAKE     = SQRT(VYW*VYW+VXW*VXW)*DT
          THENP1   = ATAN2(VYW,VXW)
          COSTHE(NP1) = COS(THENP1)
          SINTHE(NP1) = SIN(THENP1)
          X(NP2)   = X(NP1) + WAKE*COSTHE(NP1)
          Y(NP2)   = Y(NP1) + WAKE*SINTHE(NP1)
          CALL INFL  (NITR)
          CALL COEF  (SINALF,COSALF,OMEGA,UX,UY,NITR)
          CALL GAUSS(2,nt,NITR)
          CALL KUTTA (ALPHA,SINALF,COSALF,OMEGA,UX,UY)
          CALL TEWAK (SINALF,COSALF)
          TOL1     = ABS(VYW - VYWK)
          TOL2     = ABS(VXW - VXWK)
          IF ((TOL1 .LT. TOL) .AND. (TOL2 .LT. TOL)) GO TO 20
          VYW      = VYWK
          VXW      = VXWK
          NITR     = NITR + 1
          GO TO 10
   20     continue
          CALL PRESS (SINALF,COSALF,OMEGA,UX,UY,ALPHA)
          CALL PINTEG(GAMK,QK)
          if( igust .eq. 1) then
             CALL FANDM(SINAng,COSAng,cl,cd,cm)
          else
             CALL FANDM(SINALF,COSALF,cl,cd,cm)
          endif
          cla(nt) = cl
          cda(nt) = cd
          cma(nt) = cm
          ttt = ((nt-1.)*dts)/((ntmax-1.)*dts)
          IF (IRAMP .GT. 0) THEN
            write(6,1011) nt, alpha, ttt+.125, nitr, cl,cd,cm
            write(92,1011) nt, alpha, ttt+.125, nitr, cl,cd,cm
          ELSE
          write(6,1011) nt, alpha, ttt, nitr, cl,cd,cm
          write(92,1011) nt, alpha, ttt, nitr, cl,cd,cm
          ENDIF
 1011     FORMAT( i4, 2x, f7.4, 2x, f9.6, 3x, i2, 3x, 3f10.6 )
          write(8,'(2f10.5)') alpha, cl
          write(9,'(2f10.5)') alpha, cm
C..WAKE ELEMENT LEAVES TRAILING EDGE AS A CORE-VORTEX
          CV(nt)    = SS*(GAMMA-GAMK)
          XC(nt)    = X(NP1) + 0.5*WAKE*COSTHE(NP1)
          YC(nt)    = Y(NP1) + 0.5*WAKE*SINTHE(NP1)
          CVVX(nt) = VXW
          CVVY(nt) = VYW
          CALL CORVOR (SINALF,COSALF)
C..RE-INITIALISE PARAMETERS FOR NEXT TIME STEP CALCULATION
          DO 30   I = 1,NODTOT
          Q(I)     = QK(I)
          PHI(I)   = PHIK(I)
   30     CONTINUE
          GAMMA    = GAMK
          ALP      = ALPHA
          HXO      = HX
          HYO      = HY
          TOLD     = T
          DT       = TD
          T        = T + TD
```

```
      ENDDO
      if(ioscil .eq. 1 .or. itrans .eq. 1 )
     >   call PHAZ(ntmax,ntcycle,freq,itrans,alpl)
      STOP   " normal stop "
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     SUBROUTINE BODY(Z,SIGN,X,Y)                                      c
c             RETURN COORDINATES OF POINT ON THE BODY SURFACE           c
c                                                                      c
c                  Z = NODE-SPACING PARAMETER                          c
c                  X,Y = CARTESIAN COORDINATES                         c
c                  SIGN = +1. FOR UPPER SURFACE                        c
c                         -1. FOR LOWER SURFACE                        c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE BODY(Z,SIGN,X,Y)
      COMMON /PAR/ NACA,TAU,EPSMAX,PTMAX
      IF (SIGN .LT. 0.0)    Z = 1. - Z
      CALL NACA45(Z,THICK,CAMBER,BETA)
      X       = Z - SIGN*THICK*SIN(BETA)
      Y       = CAMBER + SIGN*THICK*COS(BETA)
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     SUBROUTINE COEF (SINALF,COSALF,OMEGA,UX,UY,NITR)                 c
c                                                                      c
c             SET COEFFICIENTS OF N EQUS ARISING FROM FLOW             c
c                  TANGENCY CONDITIONS AT MID POINTS OF PANELS         c
c             SOLVING THE N-SOURCE STRENGTHS IN TERMS OF THE           c
c                  VORTICITY STRENGTH (RESULTING IN 2 RHS)             c
c             KUTTA CONDITION IS SATISFIED SEPARATELY TO OBTAIN        c
c                  THE VORTICITY STRENGTH                              c
c             THIS SOLUTION METHOD IS DESIRED FOR UNSTEADY FLOW        c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE COEF (SINALF,COSALF,OMEGA,UX,UY,NITR)
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +             COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /COF/ A(201,211),NEQS
      COMMON /SING/ Q(200),GAMMA,QK(200),GAMK
      COMMON /WAK/ VYW,VXW,WAKE,DT
      COMMON /CORV/ CV(200),XC(200),YC(200),M,TD,CCVX(200),CCVY(200)
      COMMON /INF1/ AAN(201,201),BBN(201,201),AYNP1(201),BYNP1(201)
      COMMON /INF2/ SUMCCN(201),SUMCCT(201),CYNP1(200),CXNP1(200)
      COMMON /GUST/ UG(200),VG(200),XGF,UGUST,VGUST
      NEQS    = NODTOT
      NP1     = NODTOT + 1
      NP2     = NODTOT + 2
c             INITIALISE COEFFICIENTS
      IF ((M .GT. 1) .OR. (NITR .GT. 0)) GO TO 91
      DO 90   I = 1,NODTOT
      DO 90   J = 1,NP2
 90   A(I,J)  = 0.0
 91   CONTINUE
c             SET LHS MATRIX A(I,J)
      DO 120  I = 1,NODTOT
      XMID    = 0.5 * (X(I) + X(I+1))
      YMID    = 0.5 * (Y(I) + Y(I+1))
      B       = 0.0
      DO 110  J = 1,NODTOT
      IF ((M .EQ. 1) .AND. (NITR .EQ. 0)) A(I,J)  = AAN(I,J)
      B       = B + BBN(I,J)
 110  CONTINUE
c             FILL IN THE RIGHT HAND SIDE

      A(I,NP1) = -B + BBN(I,NP1)*SS/WAKE
      A(I,NP2) = -BBN(I,NP1)*GAMMA*SS/WAKE
     + + SINTHE(I)*  ((1.+UG(I))*COSALF-VG(I)*SINALF+UX)
     + - COSTHE(I)*  ((1.+UG(I))*SINALF+VG(I)*COSALF+UY)
     +               + OMEGA*(YMID*SINTHE(I) + XMID*COSTHE(I))
c        ADD CORE VORTEX CONTRIBUTION
      IF (M .EQ. 1) GOTO 140
      A(I,NP2) = A(I,NP2) - SUMCCN(I)
 140  CONTINUE
 120  CONTINUE
      RETURN
      END


c     SUBROUTINE COFISH(SINALF,COSALF)                                 c
```

```
C               SET COEFFICIENTS OF LINEAR SYSTEM - N+1 EQUATIONS        C
C                    N EQUS - FLOW TANGENCY AT MID POINTS OF PANELS      C
C                    1 EQU  - KATTA CONDITION AT TRAILING EDGE PANELS    C
C               THIS SOLUTION METHOD IS EFFECTIVE FOR STEADY FLOW, NO    C
C                    ITERATION IS REQUIRED, N-SOURCE STRENGTHS AND 1     C
C                    VORTICITY STRENGTH ARE SOLVED SIMULTANEOUSLY        C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE COFISH(SINALF,COSALF)
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +             COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /COF/ A(201,211),KUTTA
      COMMON /NUM/ PI,PI2INV
      KUTTA   = NODTOT + 1
C             INITIALISE COEFFICIENTS
      DO 90   J = 1,KUTTA
   90 A(KUTTA,J)   = 0.0
C             SET VN = 0 AT MID-POINT OF I-TH PANEL
      DO 120  I = 1,NODTOT
      XMID    = .5*(X(I) + X(I+1))
      YMID    = .5*(Y(I) + Y(I+1))
      A(I,KUTTA) = 0.0
C             --   FIND CONTRIBUTION OF J-TH PANEL
      DO 110  J = 1,NODTOT
      FLOG    = 0.0
      FTAN    = PI
      IF (J .EQ. I)     GO TO 100
      DXJ     = XMID - X(J)
      DXJP    = XMID - X(J+1)
      DYJ     = YMID - Y(J)
      DYJP    = YMID - Y(J+1)
      FLOG    = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN    = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
  100 CTIMTJ  = COSTHE(I)*COSTHE(J) + SINTHE(I)*SINTHE(J)
      STIMTJ  = SINTHE(I)*COSTHE(J) - COSTHE(I)*SINTHE(J)
      A(I,J)  = PI2INV*(FTAN*CTIMTJ + FLOG*STIMTJ)
      B       = PI2INV*(FLOG*CTIMTJ - FTAN*STIMTJ)
      A(I,KUTTA) = A(I,KUTTA) + B
      IF ((I .GT. 1) .AND. (I .LT. NODTOT))GO TO 110
C             --   IF I-TH PANEL TOUCHES TRAILING EDGE,
C                  ADD CONTRIBUTION TO KUTTA CONDITION
      A(KUTTA,J) = A(KUTTA,J) - B
      A(KUTTA,KUTTA) = A(KUTTA,KUTTA) + A(I,J)
  110 CONTINUE
C             FILL IN KNOWN SIDES
      A(I,KUTTA+1) = SINTHE(I)*COSALF - COSTHE(I)*SINALF
  120 CONTINUE
      A(KUTTA,KUTTA+1) = - (COSTHE(1) + COSTHE(NODTOT))*COSALF
     +                   - (SINTHE(1) + SINTHE(NODTOT))*SINALF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SUBROUTINE CORVOR (SINALF,COSALF)                                  C
C             COMPUTE THE LOCAL VELOCITIES OF CORE VORTICES              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CORVOR (SINALF,COSALF)
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +             COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /SING/ Q(200),GAMMA,QK(200),GAMK
      COMMON /WAK/ VYW,VXW,WAKE,DT
      COMMON /CORV/ CV(200),XC(200),YC(200),M,TD,CCVX(200),CCVY(200)
      COMMON /POT/ PHI(200),PHIK(200)
      COMMON /GUST/ UG(200),VG(200),XGF,UGUST,VGUST
      COMMON /NUM/ PI,PI2INV
      IF (M.EQ.1) GOTO 40
      MM1         = M - 1
C          VELOCITY COMPONENTS OF CORE VORTICES AT CURRENT TIME STEP
      UGC     = 0.0
      VGC     = 0.0
      DO 10   N = 1,MM1
      XG      = XC(N)*COSALF + YC(N)*SINALF
      IF (XG .GT. XGF) GO TO 5
      UGC     = UGUST
      VGC     = VGUST
    5 CONTINUE
      VY      = (1.+UGC)*SINALF+VGC*COSALF
      VX      = (1.+UGC)*COSALF-VGC*SINALF
      XMID    = XC(N)
      YMID    = YC(N)
```

```
      SUMAMY = 0.0
      SUMBMY = 0.0
C     AMY(N,J) : Y - VELOCITY INDUCED AT N-TH CORE VORTEX BY UNIT
C                STRENGTH DISTRIBUTED SOURCE ON THE J-TH PANEL
C     BMY(N,J) : Y - VELOCITY INDUCED AT N-TH CORE VORTEX BY UNIT
C                STRENGTH DISTRIBUTED VORTEX ON THE J-TH PANEL
      DO 20  J = 1,NP1
      DXJ    = XMID - X(J)
      DXJP   = XMID - X(J+1)
      DYJ    = YMID - Y(J)
      DYJP   = YMID - Y(J+1)
      FLOG   = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN   = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
      AMY    = PI2INV*(FTAN*COSTHE(J) - FLOG*SINTHE(J))
      BMY    = PI2INV*(FLOG*COSTHE(J) + FTAN*SINTHE(J))
      IF (J.EQ.NP1) GOTO 20
      SUMAMY = SUMAMY + AMY
      SUMBMY = SUMBMY + BMY
      VY     = VY + AMY*QK(J)
      VX     = VX - BMY*QK(J)
 20   CONTINUE
      VY     = VY + SUMBMY*GAMK + SS*BMY*(GAMMA-GAMK)/WAKE
      VX     = VX + SUMAMY*GAMK + SS*AMY*(GAMMA-GAMK)/WAKE
C        ADD CORE VORTEX CONTRIBUTION
C
C     CMY(N,MC) : Y - VELOCITY INDUCED AT N-TH CORE VORTEX BY UNIT
C                 STRENGTH MC-TH CORE VORTEX OTHER THAN ITSELF
C
C     CMX(N,MC) : X - VELOCITY INDUCED AT N-TH CORE VORTEX BY UNIT
C                 STRENGTH MC-TH CORE VORTEX OTHER THAN ITSELF
      DO 30  MC = 1,MM1
      IF (MC.EQ.N) GOTO 30
      DX     = XMID - XC(MC)
      DY     = YMID - YC(MC)
      DIST2  = DX*DX+DY*DY
      CMY    = -PI2INV*DX/DIST2
      CMX    = +PI2INV*DY/DIST2
      VY     = VY + CMY*CV(MC)
      VX     = VX + CMX*CV(MC)
 30   CONTINUE
C        COORDINATES OF CORE VORTICES AT NEXT TIME STEP
      CCVX(N) = VX
      CCVY(N) = VY
 10   CONTINUE
 40   CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SUBROUTINE FANDM
C             INTEGRATE PRESSURE DISTRIBUTION BY TRAPEZOIDAL RULE       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE FANDM(SINALF,COSALF,cl,cd,cm)
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +              COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /CPD/ CP(200),pivot
      CFX     = 0.0
      CFY     = 0.0
      CM      = 0.0
      DO 100  I = 1,NODTOT
c..moment coeff is computed around pivot point
      XMID    = .5*(X(I) + X(I+1)) - pivot
      YMID    = .5*(Y(I) + Y(I+1))
      DX      = X(I+1) - X(I)
      DY      = Y(I+1) - Y(I)
      CFX     = CFX + CP(I)*DY
      CFY     = CFY - CP(I)*DX
      CM      = CM + CP(I)*(DX*XMID + DY*YMID)
 100  CONTINUE
      CD      = CFX*COSALF + CFY*SINALF
      CL      = CFY*COSALF - CFX*SINALF
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SUBROUTINE GAUSS(NRHS,M,NITR)                                     C
C        SOLUTION OF LINEAR ALGEBRAIC SYSTEM BY                         C
C        GAUSS ELIMINATION WITHOUT PARTIAL PIVOTING                     C
C             (A)       = COEFFICIENT MATRIX                            C
C             NEQNS     = NUMBER OF EQUATIONS                           C
```

```
C             NRHS      = NUMBER OF RIGHT HAND SIDES                 C
C             RIGHT-HAND SIDES AND SOLUTIONS STORED IN               C
C             COLUMNS NEQNS+1 THRU NEQNS+NRHS OF (A)                 C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE GAUSS(NRHS,M,NITR)
      COMMON /COF/ A(201,211),NEQNS
      NP       = NEQNS + 1
      NTOT     = NEQNS + NRHS
      IF ((M .GT. 1) .OR. (NITR .GT. 0)) GO TO 160
C             GAUSS REDUCTION
      DO 150  I = 2,NEQNS
      IM        = I - 1
C             ELIMINATE (I-1)TH UNKNOWN FROM
C             ITH THRU (NEQNS)TH EQUATIONS
      DO 150  J = I,NEQNS
      R         = A(J,IM)/A(IM,IM)
      DO 150  K = I,NTOT
  150 A(J,K)    = A(J,K) - R*A(IM,K)
      GO TO 170
C             GAUSSIAN ELIMINATION ON ONLY THE RIGHT-HAND-SIDES
  160 DO 180  I = 2,NEQNS
      IM        = I - 1
      DO 180  J = I,NEQNS
      R         = A(J,IM)/A(IM,IM)
      DO 180  K = NP,NTOT
  180 A(J,K)    = A(J,K) - R*A(IM,K)
  170 CONTINUE
C             BACK SUBSTITUTION
      DO 220  K = NP,NTOT
      A(NEQNS,K) = A(NEQNS,K)/A(NEQNS,NEQNS)
      DO 210  L = 2,NEQNS
      I         = NEQNS + 1 - L
      IP        = I + 1
      DO 200  J = IP,NEQNS
  200 A(I,K)    = A(I,K) - A(I,J)*A(J,K)
  210 A(I,K)    = A(I,K)/A(I,I)
  220 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SUBROUTINE INDATA                                              C
C                  SET PARAMETERS OF BODY SHAPE                      C
C                  FLOW SITUATION, AND NODE DISTRIBUTION             C
C                  USER MUST INPUT                                   C
C                       NLOWER = NUMBER OF NODES ON LOWER SURFACE    C
C                       NUPPER = NUMBER OF NODES ON UPPER SURFACE    C
C                  PLUS DATA ON BODY AND SUBROUTINE BODY             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INDATA
      DIMENSION TITLE(20)
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +             COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /PAR/ NACA,TAU,EPSMAX,PTMAX
      READ (1,*) ITITLE
C     WRITE (6,*) ITITLE
      DO 10   I = 1,ITITLE
      READ (1,502) TITLE
   10 WRITE (6,503) TITLE
  501 FORMAT(3I5)
  502 FORMAT(20A4)
  503 FORMAT(1X,20A4)
      read(1,*)
      READ (1,*) IFLAG,NLOWER,NUPPER
      WRITE (6,558) IFLAG,NLOWER,NUPPER
  558 FORMAT (///2X, '=======================================',/,
     1        2X, 'IFLAG    (0:NACA, 1:INPUT)     ='      ,I5,/,
     2        2X, 'NO. PANELS UPPER SURFACE       ='      ,I5,/,
     3        2X, 'NO. PANELS LOWER SURFACE       ='      ,I5,/,
     4        2X, '=======================================')
      IF (IFLAG .NE. 0) RETURN
      read(1,*)
      READ (1,*) NACA
C     WRITE (6,501) NACA
      IEPS     = NACA/1000
      IPTMAX   = NACA/100 - 10*IEPS
      ITAU     = NACA - 1000*IEPS - 100*IPTMAX
      EPSMAX   = IEPS*0.01
      PTMAX    = IPTMAX*0.1
```

```
      TAU     = ITAU*0.01
      IF (IEPS .LT. 10) RETURN
      PTMAX   = 0.2025
      EPSMAX  = 2.6595*PTMAX**3
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SUBROUTINE INFL (NITR)                                             C
C                 CALCULATE INFLUENCE COEFFICIENTS                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INFL (NITR)
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +             COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /NUM/ PI,PI2INV
      COMMON /WAK/ VYW,VXW,WAKE,DT
      COMMON /CORV/ CV(200),XC(200),YC(200),M,TD,CCVX(200),CCVY(200)
      COMMON /INF1/ AAN(201,201),BBN(201,201),AYNP1(201),BYNP1(201)
      COMMON /INF2/ SUMCCN(201),SUMCCT(201),CYNP1(200),CXNP1(200)
      COMMON /PINTG/AANP(201,201,6),BBNP(201,201,6)
      DIMENSION PLOC(6)
      DATA PLOC/.03376524,.16939531,.38069041,
     +          .61930959,.83060469,.96623476/
      JBEG = NP1
      IF ((M .GT. 1) .OR. (NITR .GT. 0)) GO TO 510
      JBEG = 1
C     AAN(I,J) : NORMAL VELOCITY INDUCED AT MID-POINT OF I-TH PANEL
C                BY UNIT STRENGTH DISTRIBUTED SOURCE ON THE J-TH PANEL
C     BBN(I,J) : NORMAL VELOCITY INDUCED AT MID-POINT OF I-TH PANEL
C                BY UNIT STRENGTH DISTRIBUTED VORTEX ON THE J-TH PANEL
      DO 120  I = 1, NODTOT
      XMID    = .5*(X(I) + X(I+1))
      YMID    = .5*(Y(I) + Y(I+1))
      DO 110  J = 1,NODTOT
      FLOG    = 0.0
      FTAN    = PI
      IF (J .EQ. I)     GO TO 100
      DXJ     = XMID - X(J)
      DXJP    = XMID - X(J+1)
      DYJ     = YMID - Y(J)
      DYJP    = YMID - Y(J+1)
      FLOG    = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN    = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
  100 CTIMTJ  = COSTHE(I)*COSTHE(J) + SINTHE(I)*SINTHE(J)
      STIMTJ  = SINTHE(I)*COSTHE(J) - COSTHE(I)*SINTHE(J)
      AAN(I,J) = PI2INV*(FTAN*CTIMTJ + FLOG*STIMTJ)
      BBN(I,J) = PI2INV*(FLOG*CTIMTJ - FTAN*STIMTJ)
  110 CONTINUE
  120 CONTINUE
  510 CONTINUE
C   BEG COEFF POT INTEGR
      DO 122  I = 1,NODTOT
      DO 122  K = 1,6
      XMID    = X(I) + PLOC(K)*(X(I+1)-X(I))
      YMID    = Y(I) + PLOC(K)*(Y(I+1)-Y(I))
      DO 122  J = JBEG,NP1
      DXJ     = XMID - X(J)
      DXJP    = XMID - X(J+1)
      DYJ     = YMID - Y(J)
      DYJP    = YMID - Y(J+1)
      FLOG    = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN    = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
      CTIMTJ  = COSTHE(I)*COSTHE(J) + SINTHE(I)*SINTHE(J)
      STIMTJ  = SINTHE(I)*COSTHE(J) - COSTHE(I)*SINTHE(J)
      IF (I .EQ. J)  FTAN = PI
      AANP(I,J,K) = PI2INV*(FTAN*CTIMTJ + FLOG*STIMTJ)
  122 BBNP(I,J,K) = PI2INV*(FLOG*CTIMTJ - FTAN*STIMTJ)
C   END COEFF POT INTEGR
      I       = NP1
      XMID    = .5*(X(I) + X(I+1))
      YMID    = .5*(Y(I) + Y(I+1))
      DO 130  J = 1,NP1
      FLOG    = 0.0
      FTAN    = PI
      IF (J .EQ. I)     GO TO 135
      DXJ     = XMID - X(J)
      DXJP    = XMID - X(J+1)
      DYJ     = YMID - Y(J)
      DYJP    = YMID - Y(J+1)
```

```
      FLOG    = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN    = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
  135 CTIMTJ  = COSTHE(I)*COSTHE(J) + SINTHE(I)*SINTHE(J)
      STIMTJ  = SINTHE(I)*COSTHE(J) - COSTHE(I)*SINTHE(J)
      AAN(I,J) = PI2INV*(FTAN*CTIMTJ + FLOG*STIMTJ)
      BBN(I,J) = PI2INV*(FLOG*CTIMTJ - FTAN*STIMTJ)
C     AYNP1(J) : Y - VELOCITY INDUCED AT MID POINT OF WAKE ELEMENT
C                (NP1-TH PANEL) BY UNIT STRENGTH DISTRIBUTED SOURCE
C                ON J-TH PANEL
C     BYNP1(J) : Y - VELOCITY INDUCED AT MID POINT OF WAKE ELEMENT
C                (NP1-TH PANEL) BY UNIT STRENGTH DISTRIBUTED VORTEX
C                ON J-TH PANEL
      AYNP1(J) = PI2INV*(FTAN*COSTHE(J) - FLOG*SINTHE(J))
      BYNP1(J) = PI2INV*(FLOG*COSTHE(J) + FTAN*SINTHE(J))
  130 CONTINUE
      DO 140  I = 1,NODTOT
      XMID    = .5*(X(I) + X(I+1))
      YMID    = .5*(Y(I) + Y(I+1))
      J       = NP1
      DXJ     = XMID - X(J)
      DXJP    = XMID - X(J+1)
      DYJ     = YMID - Y(J)
      DYJP    = YMID - Y(J+1)
      FLOG    = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN    = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
      CTIMTJ  = COSTHE(I)*COSTHE(J) + SINTHE(I)*SINTHE(J)
      STIMTJ  = SINTHE(I)*COSTHE(J) - COSTHE(I)*SINTHE(J)
      AAN(I,J) = PI2INV*(FTAN*CTIMTJ + FLOG*STIMTJ)
      BBN(I,J) = PI2INV*(FLOG*CTIMTJ - FTAN*STIMTJ)
  140 CONTINUE
      IF (M.EQ.1) RETURN
      MM1      = M - 1
C     CYNP1(N) : Y - VELOCITY INDUCED AT MID POINT OF WAKE ELEMENT
C                (NP1-TH PANEL) BY UNIT STRENGTH N-TH CORE VORTEX
C     CXNP1(N) : X - VELOCITY INDUCED AT MID POINT OF WAKE ELEMENT
C                (NP1-TH PANEL) BY UNIT STRENGTH N-TH CORE VORTEX
      XMID    = 0.5*(X(NP1) + X(NP1+1))
      YMID    = 0.5*(Y(NP1) + Y(NP1+1))
      DO 230  N = 1,MM1
      DX      = XMID - XC(N)
      DY      = YMID - YC(N)
      DIST2   = DX*DX+DY*DY
      CYNP1(N) = -PI2INV*DX/DIST2
      CXNP1(N) = +PI2INV*DY/DIST2
  230 CONTINUE
      IF (NITR.GT.0) RETURN
C     CCN(I,J) : NORMAL VELOCITY INDUCED AT MID-POINT OF I-TH PANEL
C                BY UNIT STRENGTH N-TH CORE VORTEX
C     CCT(I,J) : TANGENTIAL VELOCITY INDUCED AT MID-POINT OF I-TH PANEL
C                BY UNIT STRENGTH N-TH CORE VORTEX
      DO 220  I = 1,NODTOT
      XMID    = 0.5*(X(I) + X(I+1))
      YMID    = 0.5*(Y(I) + Y(I+1))
      SUMCCN(I) = 0.0
      SUMCCT(I) = 0.0
      DO 210  N = 1,MM1
      DX      = XMID - XC(N)
      DY      = YMID - YC(N)
      DIST    = SQRT(DX*DX+DY*DY)
      COSTHN  = DX/DIST
      SINTHN  = DY/DIST
      CTIMTN  = COSTHE(I)*COSTHN + SINTHE(I)*SINTHN
      STIMTN  = SINTHE(I)*COSTHN - COSTHE(I)*SINTHN
      CCN     = -CTIMTN/DIST
      CCT     = -STIMTN/DIST
      SUMCCN(I) = SUMCCN(I) + CCN*CV(N)
      SUMCCT(I) = SUMCCT(I) + CCT*CV(N)
  210 CONTINUE
      SUMCCN(I) = PI2INV*SUMCCN(I)
      SUMCCT(I) = PI2INV*SUMCCT(I)
  220 CONTINUE
C  END COEFF POT INTEGR
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SUBROUTINE KUTTA (ALPHA,SINALF,COSALF,OMEGA,UX,UY)                C
C              USING KUTTA CONDITION TO DETERMINE VORTICITY             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      SUBROUTINE KUTTA (ALPHA,SINALF,COSALF,OMEGA,UX,UY)
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +             COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /COF/ A(201,211),NEQS
      COMMON /SING/ Q(200),GAMMA,QK(200),GAMK
      COMMON /WAK/ VYW,VXW,WAKE,DT
      COMMON /CORV/ CV(200),XC(200),YC(200),M,TD,CCVX(200),CCVY(200)
      COMMON /INF1/ AAN(201,201),BBN(201,201),AYNP1(201),BYNP1(201)
      COMMON /INF2/ SUMCCN(201),SUMCCT(201),CYNP1(200),CXNP1(200)
      COMMON /GUST/ UG(200),VG(200),XGF,UGUST,VGUST
      DIMENSION B1(200),B2(200),AA(2),BB(2)
C             RETRIEVE SOLUTION FROM A-MATRIX
      DO 50   I = 1,NODTOT
      B1(I)   = A(I,NP1)
   50 B2(I)   = A(I,NP2)
C             FIND VT AT TRAILING EDGE PANELS
      DO 130  K = 1,2
      IF (K .EQ. 1) I = 1
      IF (K .EQ. 2) I = NODTOT
      XMID    = 0.5 * (X(I) + X(I+1))
      YMID    = 0.5 * (Y(I) + Y(I+1))
      VTANG   =    ((1.+UG(I))*COSALF-VG(I)*SINALF+UX)*COSHE(I)
     +          + ((1.+UG(I))*SINALF+VG(I)*COSALF+UY)*SINTHE(I)
     +          + OMEGA*(YMID*COSTHE(I) - XMID*SINTHE(I))
      AA(K)   = - AAN(I,NP1)*SS/WAKE
      BB(K)   = VTANG + AAN(I,NP1)*SS*GAMMA/WAKE
      DO 120  J = 1,NODTOT
      AA(K)   = AA(K) + AAN(I,J) - BBN(I,J)*B1(J)
      BB(K)   = BB(K) - BBN(I,J)*B2(J)
  120 CONTINUE
C        ADD CORE VORTEX CONTRIBUTION
      IF (M.EQ.1) GOTO 100
      BB(K)   = BB(K) + SUMCCT(I)
  100 CONTINUE
  130 CONTINUE
C             SATISFYING KUTTA CONDITION -- SOLVE FOR VORTEX STRENGTH
      EE      = AA(1)*AA(1) - AA(2)*AA(2)
      FF      = AA(1)*BB(1) - AA(2)*BB(2) - SS/DT
      GG      = BB(1)*BB(1) - BB(2)*BB(2) + 2.*SS*GAMMA/DT
      RADI    = SQRT(FF*FF-EE*GG)
      GAMK    = (-FF - RADI)/EE
C             CALCULATE SOURCE STRENGTH
      DO 160  I = 1,NODTOT
  160 QK(I)   = GAMK*B1(I) + B2(I)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SUBROUTINE NACA45(Z,THICK,CAMBER,BETA)                          C
C             EVALUATE THICKNESS AND CAMBER                           C
C             FOR NACA 4- OR 5-DIGIT AIRFOIL                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE NACA45(Z,THICK,CAMBER,BETA)
      COMMON /PAR/ NACA,TAU,EPSMAX,PTMAX
      THICK   = 0.0
      IF (Z .LT. 1.E-10)     GO TO 100
      THICK   = 5.*TAU*(.2969*SQRT(Z) - Z*(.126 + Z*(.3537
     +          - Z*(.2843 - Z*.1015))))
  100 IF (EPSMAX .EQ. 0.0) GO TO 130
      IF (NACA .GT. 9999) GO TO 140
      IF (Z .GT. PTMAX) GO TO 110
      CAMBER  = EPSMAX/PTMAX/PTMAX*(2.*PTMAX - Z)*Z
      DCAMDX  = 2.*EPSMAX/PTMAX/PTMAX*(PTMAX - Z)
      GO TO 120
  110 CAMBER  = EPSMAX/(1.-PTMAX)**2*(1. + Z - 2.*PTMAX)*(1. - Z)
      DCAMDX  = 2.*EPSMAX/(1.-PTMAX)**2*(PTMAX - Z)
  120 BETA    = ATAN(DCAMDX)
      RETURN
  130 CAMBER  = 0.0
      BETA    = 0.0
      RETURN
  140 IF (Z .GT. PTMAX)      GO TO 150
      W       = Z/PTMAX
      CAMBER  = EPSMAX*W*((W - 3.)*W + 3. - PTMAX)
      DCAMDX  = EPSMAX*3.*W*(1. - W)/PTMAX
      GO TO 120
  150 CAMBER  = EPSMAX*(1. - Z)
      DCAMDX  = - EPSMAX
      GO TO 120
```

```
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c        SUBROUTINE PRESSs
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PRESSs
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +             COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /CPD/ CP(200),pivot
      character filnq*15,alpn*10
      alpn = '0123456789'
      filnq  = 'cps.d'
         open (unit=90,file=filnq,form='formatted')
C..Compute cp at mid point of i-th panel
         WRITE (90,'(2f12.5)')
     >      ( 0.5*(x(i)+x(i+1)), CP(I), i=1,nodtot)
      close(90)
      RETURN
      END
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     SUBROUTINE PRESS (SINALF,COSALF,OMEGA,UX,UY)                   c
c             COMPUTE UNSTEADY FLOW PRESSURE DISTRIBUTION            c
c                  AND VELOCITY POTENTIAL AT MID-POINTS OF PANELS    c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      SUBROUTINE PRESS (SINALF,COSALF,OMEGA,UX,UY,ALPHA)
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +             COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /CPD/ CP(200),pivot
      COMMON /NUM/ PI,PI2INV
      COMMON /SING/ Q(200),GAMMA,QK(200),GAMK
      COMMON /WAK/ VYW,VXW,WAKE,DT
      COMMON /CORV/ CV(200),XC(200),YC(200),M,TD,CCVX(200),CCVY(200)
      COMMON /INF1/ AAN(201,201),BBN(201,201),AYNP1(201),BYNP1(201)
      COMMON /INF2/ SUMCCN(201),SUMCCT(201),CYNP1(200),CXNP1(200)
      COMMON /POT/ PHI(200),PHIK(200)
      COMMON /GUST/ UG(200),VG(200),XGF,UGUST,VGUST
      COMMON /EXTV/ UE(200)
      COMMON /MAINout/ ialfao(20), naot, nao
      COMMON /DELPHI/ DPHITE,DPHIMP
      character filnq*15,alpn*10
      alpn = '0123456789'
c       FIND TANGENTIAL VELOCITY VT AT MID-POINT OF I-TH PANEL
      DO 130  I = 1,NODTOT
      XMID     = 0.5 * (X(I) + X(I+1))
      YMID     = 0.5 * (Y(I) + Y(I+1))
      DX       = (X(I+1) - X(I))
      DY       = (Y(I+1) - Y(I))
      DIST     = SQRT(DX*DX+DY*DY)
      VSX      = (1.+UG(I))*COSALF-VG(I)*SINALF + OMEGA*YMID + UX
      VSY      = (1.+UG(I))*SINALF+VG(I)*COSALF - OMEGA*XMID + UY
      VS       = VSX*VSX + VSY*VSY
      VTANG    =    ((1.+UG(I))*COSALF-VG(I)*SINALF+UX)*COSTHE(I)
     +            + ((1.+UG(I))*SINALF+VG(I)*COSALF+UY)*SINTHE(I)
     +            + OMEGA*(YMID*COSTHE(I) - XMID*SINTHE(I))
      VTFREE   = VTANG
C8810 DPHFRE   = DPHFRE + VTANG*DIST
C8811 DPHWKE   = DPHWKE + SS*(GAMMA-GAMK)*AAN(I,NP1)/WAKE*DIST
      VTANG    = VTANG + SS*(GAMMA-GAMK)*AAN(I,NP1)/WAKE
      DO 120  J = 1,NODTOT
      VTANG    = VTANG - BBN(I,J)*QK(J) + AAN(I,J)*GAMK
C8812 DPHGAM   = DPHGAM + AAN(I,J)*GAMK*DIST
C8813 DELPHI(J) = DELPHI(J) - BBN(I,J)*QK(J)*DIST
  120 CONTINUE
c        ADD CORE VORTEX CONTRIBUTION
      IF (M.EQ.1) GOTO 150
      VTANG    = VTANG + SUMCCT(I)
C8814 DPHWAK   = DPHWAK + SUMCCT(I)*DIST
  150 CONTINUE
      PHIK(I) = (VTANG-VTFREE)*DIST
      CP(I)    = VS - VTANG*VTANG
      UE(I)    = VTANG
  130 CONTINUE
c     COMPUTE DISTURBANCE POTENTIAL BY LINE INTEGRAL OF VELOCITY FIELD
c        INTEGRATION FROM UPSTREAM (AT INFINITY) TO THE LEADING EDGE
      NPHI     = 10 * NLOWER
      PINK     = 0.0
      XL       = 0.0
      DO 30    L = 1,NPHI
      FRACT    = FLOAT(L)/FLOAT(NPHI)
```

```
      XLP      = -10.0 * (1.0 - COS(0.5*PI*FRACT))
      DELX     = XL - XLP
      XMID     = 0.5*(XL+XLP)*COSALF
      YMID     = 0.5*(XL+XLP)*SINALF
      XL       = XLP
      VELX = UGUST
C              ADD CONTRIBUTION OF J-TH PANEL
      DO 20    J = 1,NP1
      DXJ      = XMID - X(J)
      DXJP     = XMID - X(J+1)
      DYJ      = YMID - Y(J)
      DYJP     = YMID - Y(J+1)
      FLOG     = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN     = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
      CALMTJ   = -COSALF*COSTHE(J) - SINALF*SINTHE(J)
      SALMTJ   = -SINALF*COSTHE(J) + COSALF*SINTHE(J)
      APY      = PI2INV*(FTAN*CALMTJ + FLOG*SALMTJ)
      BPY      = PI2INV*(FLOG*CALMTJ - FTAN*SALMTJ)
      IF (J .EQ. NP1) GO TO 40
      VELX     = VELX  - BPY*QK(J) +GAMK*APY
      GO TO 20
 40   VELX     = VELX  + SS*APY*(GAMMA-GAMK)/WAKE
 20   CONTINUE
C         ADD CORE VORTEX CONTRIBUTION
      IF (M .EQ. 1) GO TO 50
      MM1      = M - 1
      DO 60    N = 1,MM1
      DX       = XMID - XC(N)
      DY       = YMID - YC(N)
      DIST     = SQRT(DX*DX+DY*DY)
      COSTHN   = DX/DIST
      SINTHN   = DY/DIST
      SALMTN   = -SINALF*COSTHN + COSALF*SINTHN
      CPT      = -PI2INV*SALMTN/DIST
 60   VELX     = VELX  + CPT*CV(N)
 50   CONTINUE
      PINK     = PINK + VELX * DELX
 30   CONTINUE
C      COMPUTE DISTURBANCE POTENTIAL AT MID-POINT OF I-TH PANEL
C         LOWER SURFACE
      DO 230  I = 1,NLOWER
      PH       = -PINK
      DO 240  J = I,NLOWER
 240  PH       = PH - PHIK(J)
      PHIK(I) = PH
 230  CONTINUE
 8850 PHILOW = PHIK(1)
      DO 270  I = 1,NLOWER-1
      PHIK(I) = 0.5*(PHIK(I) + PHIK(I+1))
 270  CONTINUE
      PHIK(NLOWER) = 0.5*(PHIK(NLOWER) - PINK)
C         UPPER SURFACE
      DO 250  I = NODTOT,NLOWER+1,-1
      PH       = -PINK
      DO 260  J = NLOWER+1,I
 260  PH       = PH + PHIK(J)
      PHIK(I) = PH
 250  CONTINUE
 8851 PHIUPP = PHIK(NODTOT)
      DO 280  I = NODTOT,NLOWER+2,-1
      PHIK(I) = 0.5*(PHIK(I) + PHIK(I-1))
 280  CONTINUE
      PHIK(NLOWER+1) = 0.5*(PHIK(NLOWER+1) - PINK)
 8871 DPHITE = (PHIUPP-PHILOW)/SS
 8872 DPHIMP = (PHIK(NODTOT)-PHIK(1))/SS
         DO 290  I = 1,NODTOT
 290     CP(I)    = CP(I) - 2.*(PHIK(I)-PHI(I))/DT
      if( ( ialfao(nao) .gt. ialfao(nao-1) .and.
     >    alpha .ge. float(ialfao(nao))/10.)  .OR.
     >   ( ialfao(nao) .lt. ialfao(nao-1) .and.
     >    alpha .le. float(ialfao(nao))/10.) ) then
      itn = ialfao(nao)
      i3 = itn/100 + 1
      i2 = (itn - (i3-1)*100)/10 + 1
      i1 = (itn - (i3-1)*100 - (i2-1)*10) + 1
         if( ialfao(nao) .lt. ialfao(nao-1)) then
      filnq  = 'cpd'//alpn(i3:i3)//alpn(i2:i2)//alpn(i1:i1)//'.d'
         else
```

```
          filnq  = 'cpu'//alpn(i3:i3)//alpn(i2:i2)//alpn(i1:i1)//'.d'
             endif
             nao = nao+1
             if(nao .gt. naot) nao = 1
             open (unit=90,file=filnq,form='formatted')
C..Compute cp at mid point of i-th panel
             WRITE (90,'(2f12.6)')
     >          ( 0.5*(x(i)+x(i+1)), CP(I), i=1,nodtot)
          close(90)
          endif
          RETURN
          END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C         SUBROUTINE SETUP                                             C
C                 SETUP COORDINATES OF PANEL NODES AND SLOPES OF PANELS C
C                 COORDINATES ARE READ FROM INPUT DATA FILE UNLESS     C
C                      THE AIRFOIL IS OF NACA XXXX OR NACA 230XX TYPE  C
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
          SUBROUTINE SETUP
          COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
         +        COSTHE(201),SINTHE(201),SS,NP1,NP2
          COMMON /NUM/ PI,PI2INV
          PI      = 3.1415926585
          PI2INV  = .5/PI
C                 SET COORDINATES OF NODES ON BODY SURFACE
          IF (IFLAG .eq. 0) then
          NPOINT  = NLOWER
          SIGN    = -1.0
          NSTART  = 0
          DO 110  NSURF = 1,2
          DO 100  N = 1,NPOINT
          FRACT   = FLOAT(N-1)/FLOAT(NPOINT)
          Z       = .5*(1. - COS(PI*FRACT))
          I       = NSTART + N
          CALL BODY(Z,SIGN,X(I),Y(I))
  100     CONTINUE
          NPOINT  = NUPPER
          SIGN    = 1.0
          NSTART  = NLOWER
  110     CONTINUE
          NODTOT  = NLOWER + NUPPER
          X(NODTOT+1) = X(1)
          Y(NODTOT+1) = Y(1)
          ELSE
          NODTOT  = NLOWER + NUPPER
c         READ (1,*)  (X(I),I=1,NODTOT+1)
c         WRITE (6,501)  (X(I),I=1,NODTOT+1)
c         READ (1,*)  (Y(I),I=1,NODTOT+1)
c         WRITE (6,501)  (Y(I),I=1,NODTOT+1)
c 501     FORMAT (6F10.6)
          READ (1,*)  (X(I),Y(i),I=1,NODTOT+1)
          ENDIF
          NP1     = NODTOT + 1
          NP2     = NODTOT + 2
C.SET SLOPES OF PANELS AND CALCULATE AIRFOIL PERIMETER
          SS      = 0.0
          DO 200  I = 1,NODTOT
          DX      = X(I+1) - X(I)
          DY      = Y(I+1) - Y(I)
          DIST    = SQRT(DX*DX +DY*DY)
          SS      = SS + DIST
          SINTHE(I) = DY/DIST
          COSTHE(I) = DX/DIST
  200     CONTINUE
          RETURN
          END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C         SUBROUTINE TEWAK (SINALF,COSALF)                             C
C                 COMPUTE WAKE ELEMENT AT THE TRAILING EDGE            C
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
          SUBROUTINE TEWAK (SINALF,COSALF)
          COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
         +             COSTHE(201),SINTHE(201),SS,NP1,NP2
          COMMON /COF/ A(201,211),NEQS
          COMMON /SING/ Q(200),GAMMA,QK(200),GAMK
          COMMON /WAK/ VYW,VXW,WAKE,DT
          COMMON /WAK2/ VYWK,VXWK
          COMMON /CORV/ CV(200),XC(200),YC(200),M,TD,CCVX(200),CCVY(200)
```

```
      COMMON /INF1/ AAN(201,201),BBN(201,201),AYNP1(201),BYNP1(201)
      COMMON /INF2/ SUMCCN(201),SUMCCT(201),CYNP1(200),CXNP1(200)
      COMMON /GUST/ UG(200),VG(200),XGF,UGUST,VGUST
      XMID    = 0.5 * (X(NP1) + X(NP2))
      YMID    = 0.5 * (Y(NP1) + Y(NP2))
      UGW     = 0.0
      VGW     = 0.0
      XG      = XMID*COSALF + YMID*SINALF
      IF (XG .GT. XGF) GO TO 10
      UGW     = UGUST
      VGW     = VGUST
 10   VYWK    = (1.+UGW)*SINALF+VGW*COSALF
      VXWK    = (1.+UGW)*COSALF-VGW*SINALF
      DO 120  J = 1,NODTOT
      VYWK    = VYWK + AYNP1(J)*QK(J) + BYNP1(J)*GAMK
 120  VXWK    = VXWK - BYNP1(J)*QK(J) + AYNP1(J)*GAMK
C      ADD CORE VORTEX CONTRIBUTION
      IF (M .EQ. 1) GO TO 140
      MM1     = M - 1
      DO 130  N = 1,MM1
      VYWK    = VYWK + CYNP1(N)*CV(N)
 130  VXWK    = VXWK + CXNP1(N)*CV(N)
 140  CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     SUBROUTINE VELDIS(SINALF,COSALF)                                  C
C             COMPUTE STEADY FLOW PRESSURE DISTRIBUTION                 C
C                  AND VELOCITY POTENTIAL AT MID-POINTS OF PANELS       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE VELDIS(SINALF,COSALF)
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +             COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /COF/ A(201,211),KUTTA
      COMMON /CPD/ CP(200),pivot
      COMMON /NUM/ PI,PI2INV
      COMMON /SING/ Q(200),GAMMA,QK(200),GAMK
      COMMON /POT/ PHI(200),PHIK(200)
      COMMON /GUST/ UG(200),VG(200),XGF,UGUST,VGUST
      COMMON /EXTV/ UE(200)
 8870 COMMON /DELPHI/ DPHITE,DPHIMP
C             RETRIEVE SOLUTION FROM A-MATRIX
      DO 50   I = 1,NODTOT
 50   Q(I)    = A(I,KUTTA+1)
      GAMMA   = A(KUTTA,KUTTA+1)
C             FIND VT AND CP AT MID-POINT OF I-TH PANEL
      DO 130  I = 1,NODTOT
      XMID    = .5*(X(I) + X(I+1))
      YMID    = .5*(Y(I) + Y(I+1))

**    VTANG  >>>  V/V(inf)
      VTANG   = COSALF*COSTHE(I) + SINALF*SINTHE(I)
      VTFREE  = VTANG
C             ADD CONTRIBUTION OF J-TH PANEL
      DO 120  J = 1,NODTOT
      FLOG    = 0.0
      FTAN    = PI
      IF (J .EQ. I) GO TO 100
      DXJ     = XMID - X(J)
      DXJP    = XMID - X(J+1)
      DYJ     = YMID - Y(J)
      DYJP    = YMID - Y(J+1)
      FLOG    = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN    = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
 100  CTIMTJ  = COSTHE(I)*COSTHE(J) + SINTHE(I)*SINTHE(J)
      STIMTJ  = SINTHE(I)*COSTHE(J) - COSTHE(I)*SINTHE(J)
      AA      = PI2INV*(FTAN*CTIMTJ + FLOG*STIMTJ)
      B       = PI2INV*(FLOG*CTIMTJ - FTAN*STIMTJ)
      VTANG   = VTANG - B*Q(J) +GAMMA*AA
 120  CONTINUE
      CP(I)   = 1.0 - VTANG*VTANG
      UE(I)   = VTANG
      write (91,*)xmid,abs(vtang)
      WRITE (6,1050) I,XMID,Q(I),GAMMA,CP(I),UE(I)
C     WRITE (19,*) XMID,-CP(I)
C        INITIAL SET-UP FOR DISTURBANCE POTENTIAL CALCULATION
      DX      = X(I+1) - X(I)
      DY      = Y(I+1) - Y(I)
```

```
      DIST     = SQRT(DX*DX+DY*DY)
      PHI(I)   = (VTANG-VTFREE)*DIST
  130 CONTINUE
C     COMPUTE DISTURBANCE POTENTIAL BY LINE INTEGRAL OF VELOCITY FIELD
C        INTEGRATION FROM UPSTREAM (AT INFINITY) TO THE LEADING EDGE
      NPHI     = 10 * NLOWER
      PIN      = 0.0
      XL       = 0.0
      DO 30    L = 1,NPHI
      FRACT    = FLOAT(L)/FLOAT(NPHI)
      XLP      = -10.0 * (1.0 - COS(0.5*PI*FRACT))
      DELX     = XL - XLP
      XMID     = 0.5*(XL+XLP)*COSALF
      YMID     = 0.5*(XL+XLP)*SINALF
      XL       = XLP
      VELX     =  UGUST
C              ADD CONTRIBUTION OF J-TH PANEL
      DO 20    J = 1,NODTOT
      DXJ      = XMID - X(J)
      DXJP     = XMID - X(J+1)
      DYJ      = YMID - Y(J)
      DYJP     = YMID - Y(J+1)
      FLOG     = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN     = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
      CALMTJ   = -COSALF*COSTHE(J) - SINALF*SINTHE(J)
      SALMTJ   = -SINALF*COSTHE(J) + COSALF*SINTHE(J)
      APY      = PI2INV*(FTAN*CALMTJ + FLOG*SALMTJ)
      BPY      = PI2INV*(FLOG*CALMTJ - FTAN*SALMTJ)
      VELX     = VELX  - BPY*Q(J) +GAMMA*APY
   20 CONTINUE
      PIN      = PIN + VELX * DELX
   30 CONTINUE
C      COMPUTE DISTURBANCE POTENTIAL AT MID-POINT OF I-TH PANEL
C        LOWER SURFACE
      DO 230   I = 1,NLOWER
      PH       = -PIN
      DO 240   J = I,NLOWER
  240 PH       = PH - PHI(J)
      PHI(I)   = PH
  230 CONTINUE
 8861 PHILOW = PHI(1)
      DO 270   I = 1,NLOWER-1
      PHI(I)   = 0.5*(PHI(I) + PHI(I+1))
  270 CONTINUE
      PHI(NLOWER) = 0.5*(PHI(NLOWER) - PIN)
C        UPPER SURFACE
      DO 250   I = NODTOT,NLOWER+1,-1
      PH       = -PIN
      DO 260   J = NLOWER+1,I
  260 PH       = PH + PHI(J)
      PHI(I)   = PH
  250 CONTINUE
 8860 PHIUPP = PHI(NODTOT)
      DO 280   I = NODTOT,NLOWER+2,-1
      PHI(I)   = 0.5*(PHI(I) + PHI(I-1))
  280 CONTINUE
      PHI(NLOWER+1) = 0.5*(PHI(NLOWER+1) - PIN)
 1000 FORMAT(/,4X,'J',4X,'X(J)',6X,'Q(J)',5X,'GAMMA',5X,
     + 'CP(J)',6X,'V(J)',/)
 1050 FORMAT(I5,5F10.6)
 8871 DPHITE = (PHIUPP-PHILOW)/SS
 8872 DPHIMP = (PHI(NODTOT)-PHI(1))/SS
C8862 WRITE (8,8863) (PHIUPP-PHILOW)/SS
C8863 FORMAT (//1X,'CIRCULATION VIA POTENTIAL:',E14.6//)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PINTEG(GAMMA,Q)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      COMMON /BOD/ IFLAG,NLOWER,NUPPER,NODTOT,X(202),Y(202),
     +             COSTHE(201),SINTHE(201),SS,NP1,NP2
      COMMON /NUM/ PI,PI2INV
      COMMON /DELPHI/ DPHITE,DPHIMP
      COMMON /PINTG/AANP(201,201,6),BBNP(201,201,6)
      DIMENSION Q(200)
      DIMENSION WGHT(6)
      DATA WGHT/.08566225,.18038079,.23395697,
     +          .23395697,.18038079,.08566225/
```

```
C *** PRECISE CONTOUR INTEGRATION ****
      SUMC = 0.0
      DO 8000  I = 1,NODTOT
      DX       = (X(I+1) - X(I))
      DY       = (Y(I+1) - Y(I))
      DIST     = SQRT(DX*DX+DY*DY)
      DO 8000  K = 1,6
      VTANG    = 0.0
      DO 8100  J = 1,NODTOT
 8100 VTANG    = VTANG - BBNP(I,J,K)*Q(J) + AANP(I,J,K)*GAMMA
 8000 SUMC = SUMC + VTANG*DIST*WGHT(K)
C *** DUMMY INTEGRATION ****
      SUM = 0.0
C  INTEGRATION FROM TRAILING EDGE TO (1.0,-0.1)
      XMID = 1.0
      Y1 = 0.0
      DO 9100 K=1,10
      Y2 = -FLOAT(K)/100.
      YMID = 0.5*(Y1+Y2)
      DELY = Y2-Y1
      SUM1 = 0.0
      DO 9000 J = 1,NODTOT
      DXJ      = XMID - X(J)
      DXJP     = XMID - X(J+1)
      DYJ      = YMID - Y(J)
      DYJP     = YMID - Y(J+1)
      FLOG     = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN     = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
      APY      = PI2INV*(FTAN*COSTHE(J) - FLOG*SINTHE(J))
      BPY      = PI2INV*(FLOG*COSTHE(J) + FTAN*SINTHE(J))
 9000 SUM1     = SUM1 + APY*Q(J) + BPY*GAMMA
      SUM      = SUM + SUM1*DELY
      Y1 = Y2
 9100 CONTINUE
C  INTEGRATION FROM (1.0,-0.1) TO (-0.1,-0.1)
      YMID = -0.1
      X1 = 1.0
      DO 9200 K=1,100
      X2 = 1.0-1.1*FLOAT(K)/100.
      XMID = 0.5*(X1+X2)
      DELX = X2-X1
      SUM1 = 0.0
      DO 9250 J = 1,NODTOT
      DXJ      = XMID - X(J)
      DXJP     = XMID - X(J+1)
      DYJ      = YMID - Y(J)
      DYJP     = YMID - Y(J+1)
      FLOG     = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN     = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
      APY      = PI2INV*(FTAN*COSTHE(J) - FLOG*SINTHE(J))
      BPY      = PI2INV*(FLOG*COSTHE(J) + FTAN*SINTHE(J))
 9250 SUM1     = SUM1 - BPY*Q(J) + APY*GAMMA
      X1 = X2
      SUM      = SUM + SUM1*DELX
 9200 CONTINUE
C  INTEGRATION FROM (-0.1,-0.1) TO (-0.1,0.1)
      XMID = -0.1
      Y1 = -0.1
      DO 9300 K=1,20
      Y2 = -0.1+FLOAT(K)/100.
      YMID = 0.5*(Y1+Y2)
      DELY = Y2-Y1
      SUM1 = 0.0
      DO 9350 J = 1,NODTOT
      DXJ      = XMID - X(J)
      DXJP     = XMID - X(J+1)
      DYJ      = YMID - Y(J)
      DYJP     = YMID - Y(J+1)
      FLOG     = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN     = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
      APY      = PI2INV*(FTAN*COSTHE(J) - FLOG*SINTHE(J))
      BPY      = PI2INV*(FLOG*COSTHE(J) + FTAN*SINTHE(J))
 9350 SUM1     = SUM1 + APY*Q(J) + BPY*GAMMA
      SUM      = SUM + SUM1*DELY
      Y1 = Y2
 9300 CONTINUE
C  INTEGRATION FROM (-0.1,0.1) TO (1.0,0.1)
      YMID = 0.1
```

```
      X1 = -0.1
      DO 9400 K=1,100
      X2 = -0.1+1.1*FLOAT(K)/100.
      XMID = 0.5*(X1+X2)
      DELX = X2-X1
      SUM1 = 0.0
      DO 9450 J = 1,NODTOT
      DXJ      = XMID - X(J)
      DXJP     = XMID - X(J+1)
      DYJ      = YMID - Y(J)
      DYJP     = YMID - Y(J+1)
      FLOG     = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN     = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
      APY      = PI2INV*(FTAN*COSTHE(J) - FLOG*SINTHE(J))
      BPY      = PI2INV*(FLOG*COSTHE(J) + FTAN*SINTHE(J))
 9450 SUM1     = SUM1 - BPY*Q(J) + APY*GAMMA
      X1 = X2
      SUM      = SUM + SUM1*DELX
 9400 CONTINUE
C  INTEGRATION FROM (1.0,0.1) TO TRAILING EDGE
      XMID = 1.0
      Y1 = 0.1
      DO 9500 K=1,10
      Y2 = 0.1-FLOAT(K)/100.
      YMID = 0.5*(Y1+Y2)
      DELY = Y2-Y1
      SUM1 = 0.0
      DO 9550 J = 1,NODTOT
      DXJ      = XMID - X(J)
      DXJP     = XMID - X(J+1)
      DYJ      = YMID - Y(J)
      DYJP     = YMID - Y(J+1)
      FLOG     = .5*ALOG((DXJP*DXJP+DYJP*DYJP)/(DXJ*DXJ+DYJ*DYJ))
      FTAN     = ATAN2(DYJP*DXJ-DXJP*DYJ,DXJP*DXJ+DYJP*DYJ)
      APY      = PI2INV*(FTAN*COSTHE(J) - FLOG*SINTHE(J))
      BPY      = PI2INV*(FLOG*COSTHE(J) + FTAN*SINTHE(J))
 9550 SUM1     = SUM1 + APY*Q(J) + BPY*GAMMA
      SUM      = SUM + SUM1*DELY
      Y1 = Y2
 9500 CONTINUE
c9600 WRITE (6,9660) GAMMA,DPHITE,DPHIMP,SUM/SS,SUMC/SS
c9660 FORMAT (//1X,52(1H=)//,
c    +           1X,'C O M P A R I S O N   O F   G A M M A S'//,
c    +           1X,'GAMMA FROM KUTTA CONDITION:             ',F12.8/
c    +           1X,'GAMMA FROM CONTOUR INTEGR (TRAIL EDGE): ',F12.8/
c    +           1X,'GAMMA FROM CONTOUR INTEGR (MIDPOINTS):  ',F12.8/
c    +           1X,'GAMMA FROM BOX INTEGR (OFF THE CONTOUR):',F12.8/
c    +           1X,'GAMMA FROM PRECISE CONTOUR INTEG (6 PT):',F12.8//
c    +           1X,52(1H=)//)
      RETURN
      END
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
C  THIS PROGRAM TAKES THE INPUT FILE (FILE CODE 14) AND CONVERTS THE
C  DATA TO A FREQUENCY, AMPLITUDE, AND PHASE SHIFT.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PHA2(npts,ntcycle,w,itrans,alp1)
      parameter (nwmx=200, npmx=201)
      DIMENSION PHASE(3), AMP(3),CKT(400),
     >         FN(nwmx),R(nwmx),DAT(nwmx),FNT(3,nwmx)
       LOGICAL flag
       REAL L1,L2,L3,L4,M1,M2,M3,M4
       common /phase   t(nwmx),alpha(nwmx),cl(nwmx),cd(nwmx),
     >                 cm(nwmx),hy(nwmx)
       print*, '  '
       print*, '  '
       print*, '                PHASE SHIFT ANALYSIS    '
       PI = ACOS(-1.0)
       do i=1,3
       amp(I)=0.0
       end do
       DO 200, J = 1,2
          DO I = 1,NPTS
             IF (J .lt. 1.5) THEN
                DAT(I) = CL(I)
             ELSE IF(J .gt.1.5) THEN
                DAT(I) = CM(I)
             END IF
          END DO
```

```
      C  READ POSITION DATA
           IF(itrans .EQ. 1) THEN
            DO I=1,NPTS
               alpha(I) = HY(I)
            END DO
            zero = .00001
           ELSE
            zero = .01
           END IF
           CALL AMPLITUDE(DAT,AMP,NPTS,J)
      c..DETERMINE PHASE SHIFT
           PHI = 0.
           ERR = 10000.
           CN = -2.0
           itter = 500
           iCOUNT  = 0
           phi = cn*pi/180.0
           nts = npts - 3*ntcycle/4
           nte = npts -   ntcycle/4
      C..BEGIN ITTERATION TO CONVERGENCE
       30  iCOUNT = iCOUNT + 1
           SUM = 0
           DO I = nts, nte
              FN(I) = -AMP(J)*cos(W*T(I) + PHI )
              R(I) = ABS(FN(I) - DAT(I))
              SUM = SUM + R(I)
           END DO
      c    print*, 'icount, phi, cn err :',icount,phi*180./pi,cn,err
           IF(sum .gt. err) THEN
             CN = -0.5*CN
           endif
           PHI = PHI + CN*PI/180.0
           ERR = SUM
           IF( abs(cn) .gt. 0.001 .and. icount .lt. itter ) GO TO 30
           PHASE(J) = PHI*180.0/PI

      c    do i = 4,npts
      c       FNT(J,I) = AMP(J)*SIN(W*T(I) + PHASE(J)*pi/180.0)
      c    end do
       200 CONTINUE
          open (unit=15,file='phase.d',form='formatted')
          write(15,'(4f12.5)')
      c  >    ( t(i),alpha(i), fnt(1,i),fnt(2,i), I=1,npts)
         >    ( t(i),alpha(i), cl(i),cm(i), I=1,npts)
          close(15)
            print*, '  '
            print*, 'AMPLITUDE;  clamp, cmamp :',amp(1),amp(2)
            print*, 'PHASE;       clp,    cmp   :',
         >          phase(1)+180, phase(2)
      c  DETERMINE THE PROPULSIVE EFFICIENCY
           PHASE(1) = PHASE(1)*pi/180.0
           PHASE(2) = PHASE(2)*pi/180.0
           CDTOT = 0
           k = 0
           DO I =npts-ntcycle, npts
            CDTOT = CDTOT + CD(I) - CD(1)
            k = k+1
           END DO
           DBAR = CDTOT/K
           DBAR = DBAR
           PRINT*,'AVERAGE DRAG, TOTAL DRAG : ', DBAR,CDTOT
           IF(itrans .EQ. 1) THEN
             WBAR = -.5*w*SIN(PHASE(1))*AMP(1)
             ETA  = 2*DBAR/WBAR
           ELSE
             WBAR = .5*w*SIN(PHASE(2))*AMP(2)
             ETAS = DBAR/WBAR
           END IF
           PRINT*,'ETAS, WBAR              : ',ETAS,WBAR
      C  DETERMINE AERODYNAMIC FORCES
           PHASE(1) = PHASE(1) + PI
           AMP(1) = AMP(1)/2.0
            IF(itrans .EQ. 1) THEN
              L1=AMP(1)*cos(PHASE(1))/(pi*(w/2.0)**2*alp1)
              L2=AMP(1)*sin(PHASE(1))/(pi*(w/2.0)**2*alp1)
              M1= .5
              M2=  0
      c        print*,'L1,L2 = ',L1,L2,amp(1)
```

```
c          print*,'M1,M2 = ', M1,M2,amp(2)
       ELSE
c          print*,'INPUT L1, L2 ='
c          read(*,*) L1,L2
c          L3=2*AMP(1)*cos(PHASE(1))/(pi*(w/2.0)**2*alp1) + .5*L1
c          L4=2*AMP(1)*sin(PHASE(1))/(pi*(w/2.0)**2*alp1) + .5*L2
         M1 = .5
         M2 = 0
         M3 = .375
         M4 = -(2/W)
        END IF
        return
        end
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       SUBROUTINE AMPLITUDE(DAT,AMP,NPTS,J)
       DIMENSION DAT(200),AMP(3),AMP1(10),AMP2(10)
         n2 = 0
         m2 = 0
         do i= 1,10
            amp1(i) = 0
            amp2(i) = 0
         end do
         DO I = 2,NPTS-1
        IF(DAT(I+1) .LT. 0 .AND. DAT(I) .LT. 0 ) THEN
            IF(ABS(DAT(I+1)) .GT.ABS(DAT(I))) then
            if((n2+1)/2.0 .EQ. int((n2+1)/2.0))  n2 = n2+1
              TMP = ABS(DAT(I+1))
            else
              goto 10
            end if
            if(tmp .gt. amp1(n2)) then
              amp1(n2) = tmp
              tmp =0
            end if
         ELSE
            if((n2+1)/2.0 .ne. int((n2+1)/2.0))  n2 = n2+1
            IF(ABS(DAT(I+1)) .GT.ABS(DAT(I))) THEN
               TMP = ABS(DAT(I+1))
            else
              goto 10
            end if
            if(tmp .gt. amp2(n2)) then
              amp2(n2) = tmp
              tmp =0
            end if
          end if
 10       END DO
          if (amp1(2) .gt. amp2(2)) then
            if(amp1(2) .lt. amp2(3)) then
               comp = amp1(2)
            else
               comp = amp2(3)
            end if
          Else
            if(amp2(2) .lt. amp1(3)) then
                 comp = amp2(2)
            else
                 comp = amp1(3)
            end if
           end if
      do i = 2,n2
        if(abs(amp1(I)-Comp) .GT. .1*comP) go to 20
          m2 = m2 + 1
          amp(j) =  amp(j) +amp1(i)
          go to 30
 20      if(abs(amp2(I)-comP) .GT. .1*comP) go to 30
          m2 = m2 +1
          amp(j) = amp(j) + amp2(i)
 30   continue
      end do
      if (j .lt. 3) then
         AMP(J) = AMP(J)/(M2)
      else
         amp(j) = amp(j)/(2*m2)
      end if
      return
      END
```

APPENDIX C

A. NS.IN NAME LIST

```
**********************************************************
No dphi/dt, circulation is applied
213x61 new expanding smooth grid....ITEU=183...ITEL=31
----------------------------------------------------------

C.. IREAD,   ITER    NPRINT,   NLOAD  ODALFA
     0       2000     100       500     1.0
C.. POTEN, NTPOT,  MPOT,   MDF   KSISO, SO_DIST  ← (Line not used)
    false    1       1      1     4      0.15
C.. ALPHA    OSCIL    RAMP   REDFRE  ALFAMND  ALFAMXD
    2.00     false    false   0.01    0.001    20.0
C.. MACH   REYNOLD    VISC     TURBL
   0.200   2000000.   true     true
C.. TIMEACC    COUR       NEWTIT
    false      100          2

------------------------------------------------------------------
```

..comments..

IREAD: 0: no initial solution
1: initial solution binary ** (cp ends.d strs.d)
2: initial solution formatted (plot3d form)

ITER: # of iterations..
Ramp: $\{(\alpha_{max} - \alpha_{min})/(2\ A\ M\ dtau)\}$
Sinusoid: $\{\Pi\ /(k\ M\ dtau)\}$

NPRINT: prints residuals at every nprint timesteps

NLOAD: prints loads at every nload timesteps

OALFA: prints out q file at every oalpha degree

- poten: false: no interactive solution
  true: potential flow ns interavtive solution (Inner and Outer Grid)
- ntpot: timestep where interavtive solution starts
- mpot: interactive boundary conditons are updated at every mpot timestep
- mdf: dphi/dt is computed at every mdf timestep
- ksiso: inner boundary is located ksiso grid points inside the outer boundary
- so_dist: where the outer boundary is located (in terms of chord lengths

ALPHA: steady state aoa

OSCIL: false: no sinuzaidal oscillations
true: sinuzaidal oscillations

RAMP: false: no straight ramp motion
true: straight ramp motion

REDFRE: Reduced frequency based on HALF CHORD, chord length is assumed to be 1.

ALFAMND: starting or min aoa

ALFAMXD: max aoa

VISC: false: euler solution
true: ns solution

TURBL: false: laminar flow
true: b-1 turbulence model

TIMEACC: false: variable time stepping ** steady state, Attached flow
true: constant time stepping ** Ramp or Sinusoids

COUR: courant number of the timestepping 50-200-1000

NEWTIT: Number of newton subiterations in each timestep, applied in unsteady flows. (2-3)

B. BATCH & GRAPHICS INTERFACE CODES

****
RUN

```
# Indigo batch for executables
#! /bin/csh -f
#Runs ns.f versions and postprocesses for INDIGO machines
nohup
#
set SRC=~/ns/src
set NS=ns
set NO=no
#
if ($#argv == 0) then
  echo "   "
  echo "•  MISSING argument :  s, wp or cl...."
else if ($1 == "s") then
  echo "   "
  echo "     RUNNING -ns- background..."
  echo "   "
  echo "     " > $NO ; date >> $NO ; echo "     " >> $NO ; \
  cat ns.in >> $NO ;  \
  (timex nice -3 $SRC/$NS < ns.in >>& $NO) >> $NO.t ;echo " " >> $NO ;\
  cat $NO.t >> $NO ; date >> $NO ; \
  echo "•  ns in $cwd has RUN.."  ; rm $NO.t &
else if ($1 == "wp") then
   if ($#argv != 2) then
      echo "   "
      echo " Missing the INPUT file argument.. "
   else
      echo "   "
      echo " Writing PLOT3D files.."
      mv $2 inf; $SRC/wp3d ; mv inf $2 ;echo "• Written.." &
   endif
else if ($1 == "cl") then
   $SRC/wrcl
else
   echo "   "
   echo "^G  WRONG argument, try   s, wp or cl...."
endif
```
****

RUNS

```
C... Cray batch executable
c       Submits a batch request to cray unicos network queing system
#QSUB -q prem -lT 99:59:59 -lM 64Mw
set SRC=ns/src
set NS=ns
set NO=ns/no
```

```
echo "   " > $NO ; date >> $NO ; echo "   " >> $NO ; \
cat ns/ns.in >> $NO ; \
( $SRC/$NS < ns/ns.in >> $NO ) >> $NO.t ; echo "   " >> $NO ; \
cat $NO.t >> $NO ; date >> $NO ; \
exit 0
*******************************************************************
```

WP3D.F

```
      PROGRAM WP3D
C..Reads binary last iteration file and writes it in plot3d format with Rotated
C...   Grid
      parameter (imax=250,kmax=100)
      dimension   q(4,imax,kmax),x(imax,kmax),z(imax,kmax)
     >                         ,xr(imax,kmax),zr(imax,kmax)
      character filnq*15,alpn*10,filngr*15
      alpn = '0123456789'
      pi = 3.14159
c..read the grid
      open (7,file='/d3/johnston/ns/grid.in',status='old',
     +                                  form='formatted')
      read (7,*) imx, kmx, iws, iwe
      read (7,*) ((x(i,k), i = 1, imx), k = 1, kmx ),
     >           ((z(i,k), i = 1, imx), k = 1, kmx )
      close(7)
      open (8,file='inf',status='old',form='unformatted')
      DO 100 II = 9,100,2
      read (8,end=101) imx, kmx
      read (8) amach,alfad,reynph,time,itn
      read (8) ((( q(l,i,k), i=1,imx), k=1,kmx), l=1,4)
C..extract character string for iteration counT
         i6 = itn/100000 + 1
         i5 = (itn - (i6-1)*100000)/10000 + 1
         i4 = (itn - (i6-1)*100000 - (i5-1)*10000)/1000 + 1
         i3 = (itn - (i6-1)*100000 - (i5-1)*10000
     >                             - (i4-1)*1000)/100 + 1
         i2 = (itn - (i6-1)*100000 - (I5-1)*10000 - (I4-1)*1000
     >             - (i3-1)*100 )/10 + 1
         i1 = (itn - (i6-1)*100000 - (i5-1)*10000 - (I4-1)*1000
     >             - (i3-1)*100    - (i2-1)*10 ) + 1

         filnq  = 'q'//alpn(i5:i5)//alpn(i4:i4)//alpn(i3:i3)//
     >                alpn(i2:i2)//alpn(i1:i1)//'.fmt'
         filngr = 'gr'//alpn(i5:i5)//alpn(i4:i4)//alpn(i3:i3)//
     >                alpn(i2:i2)//alpn(i1:i1)//'.fmt'
c..write the qfile
        open (ii,file=filnq,form='formatted')
        write(ii,*) imx,kmx
        write(ii,'(5e15.7)') amach,alfad,reynph,time
        write(ii,'(5e15.7)')
     >   ((( q(is,i,k),i=1,imx ),k=1,kmx ), is=1,4)
        close(ii)
c..write the rotated grid
        dalfa   = alfad * pi/180.
        ca = cos( dalfa )
        sa =-sin( dalfa )
        do 10 i=1,imx
        do 10 k=1,kmx
        xr(i,k) = x(i,k) * ca - z(i,k) * sa
        zr(i,k) = z(i,k) * ca + x(i,k) * sa
   10   continue
        ig = ii + 1
        open (ig,file=filnGR,form='formatted')
        write(ig,*) imx,kmx, iws, iwe
        write(ig,'(5e15.7)')
     >         ((xr(i,k), i = 1, imx), k = 1, kmx ),
     >         ((zr(i,k), i = 1, imx), k = 1, kmx )
        close(ig)
  100 Continue
  101 close(8)
      STOP
      END

*******************************************************************
```

WRCL.F

```
       PROGRAM wrcl
*** auto-writes all files given dtau and niter
*** includes Cf data
```

```
*** Corrects NS.F grid/load error
C..write alpha and cl from loads.d file
      parameter (idim=215 )
      dimension cp(idim,idim), cl(idim),cd(idim),cm(idim),time(idim),
     >   alpha(idim),clv(idim),cdv(idim),cmv(idim),clw(idim),cdw(idim),
     >   cmw(idim), x(idim,idim),z(idim,idim),cf(idim,idim)
      character fname*80
*         Print*,'input niter = '
*         read*,niter
*         print*,'input dtau = '
*         read*,dtau
      print*,' '
      print*,'  Enter LOAD file name  :>'
      read(*,'(a80)') fname
      open(2,file=fname,form='formatted',status='old')
      do it = 1, 1000
      read(2,*,end=21)iter,alpha(it),time(it),fsmach,re,itel,iteu,
     >             cl(it),cd(it),cm(it), clv(it),cdv(it),cmv(it),
     >             (cp(i), i=1,iteu-itel+1),
     >             (cf(i), i=1,iteu-itel+1)
      enddo
   21 close(2)
      open(11,file='/d3/johnston/ns/grid.in',form='formatted',
     +             status='old')
       read (11,*) imx, kmx, iwks, iwke
       read (11,*) ((x(i,k), i = 1, imx), k = 1, kmx ),
     >             ((z(i,k), i = 1, imx), k = 1, kmx )
      close(11)
-----------------------------------------------------
        do 30 itt = 1,it-1
        cn=0.
        ch=0.
        cmp=0.
        ralfa = alpha(itt)*pi/180.
        do 25 i=itel+1,iteu
        dx = x(i+1) - x(i-1,1)
        dz = z(i,1) - z(i-1,1)
        avcp = .5*( cp(i,itt) + cp(i-1,itt) )
        cn = cn - avcp*dx
        ch = ch + avcp*dz
        zave = .5*(z(i,1) + z(i-1,1) )
        xave = .5*(x(i,1) + x(i-1,1) )
C.. Cm about .25 x/c
        cmp = cmp + avcp*( dz*zave + dx*(xave-.25) )
 25     continue
        clw(itt) = cn*cos(ralfa) - ch*sin(ralfa)
        cdw(itt) = cn*sin(ralfa) + ch*cos(ralfa)
        cmw(itt) = cmp
 30     continue

-------------------------------------------------------
        open(3,file='cla.d',form='formatted')
        write(3,'(2f10.5)') ( alpha(i), -clw(i), i = 1,it-1)

*        open(4,file='clt.d',form='formatted')
*        write(4,'(2f10.5)') (time(i)/niter/dtau, -clw(i), i = 1,it-1)

        open(5,file='cda.d',form='formatted')
        write(5,'(2f10.5)') ( alpha(i), -cdw(i), i = 1,it-1)

*        open(6,file='cdt.d',form='formatted')
*        write(6,'(2f10.5)') (time(i)/niter/dtau, -cdw(i), i = 1,it-1)

        open(7,file='cma.d',form='formatted')
        write(7,'(2f10.5)') ( alpha(i), -cmw(i), i = 1,it-1)

*        open(8,file='cmt.d',form='formatted')
*        write(8,'(2f10.5)') (time(i)/niter/dtau, -cmw(i), i = 1,it-1)

        open(9,file='cp.d',form='formatted')
        write(9,'(2f10.5)') ( x(i,1), -cp(i,it-1), i = itel,iteu)

        open(10,file='cf.d',form='formatted')
        write(10,'(2f10.5)') ( x(i,1), cf(i,it-1), i = itel,iteu)
      STOP
      END

***********************************************************************
```

PLCON.F

```
      PROGRAM PLCON
C..reads gr.. and q .. solution files and plots contours
C..MUST COMPILE AND LINK THIS PROGRAM
**  f77 -c plcon.f
**  f77 plcon.o /usr/local/bin/nasadig.a -o plcon
      parameter (idim=213 ,kdim=61 )
      dimension  q(idim,kdim,4), xy(idim,kdim,2), ax(idim),ay(idim)
     >            ,ybl(kdim), vbl(kdim), func(idim,kdim)
      character yn*1,title*80,fname*80, text*80
      integer*4 ititle(20)
      equivalence (title, ititle(1) )
      data xumin,xumax,yumin /-0.75, 1.50, -0.75 /
c      data xumin,xumax,yumin /-0.1, 1.25, -0.35 /
     >    ,xlen,ylen/3.25,  2.5 /
     >    ,gam /1.4/

C..nasadig calls
       print*,'  Enter Out.PS file name :>'
        read(*,102) FNAME
      open(10, file = fname,form='formatted',status='unknown')
      CALL postsc(10)
c     call device(ktype,xpage,ypage)
      CALL PAGE(11., 8.5)
c     CALL HRDrot('COMIC')
c     CALL HRDrot('MOVIE')
      CALL HRDSCL('NONE')
      CALL NOBord
      CALL NOCHEK
      CALL DUPLex
      CALL HEIGHT(0.08)
      CALL frmwid(0.012)
      CALL crvwid(0.001)
      CALL MARGIN(0.)
       print*,' Input Ramp=1 or Sinusoid=2 or SS=9'
       read*,jj
CC     View area of interest definition
        WRITE(*,105) XUMIN,XUMAX,yumin
  105  FORMAT(/'   xumin, xumax, yumin   : ',
     >         '(', 3F7.2,' )','   Redefine ? (n):>'  )
       READ(*,101) YN
       IF (YN.EQ.'Y' .OR.  YN.EQ.'y') then
        print*,'Input   xumin, xumax, yumin '
        READ(*,*) XUMIN,XUMAX,yumin
       endif
C..read caption  (Placed at bottom of page in Landscape mode)
c      print*,'  Any caption for the plots ?.   (caption) :>'
c      read(*,102,END=10) TITLE
  102 format(A80)
c      TITLE= ' '
C..read  grid
      alphao = 0.
           alphao = 0.
c      FNAME = 'grid.in'
       print*,'  '
c      print*,'  Default grid file is grid.in  Is this o.k.? (y or n) '
c      read(*,101) yn
  101 format(a1)
c      if( yn .eq. 'n' .or. yn .eq. 'N') then
   13   print*,'  Enter GRID file name :>'
        read(*,102) FNAME
c      endif
        OPEN(1,FILE=FNAME,FORM='FORMATTED',STATUS='OLD')
        read(1,*) imx,kmx,iws,iwe
        read(1,*)  ((xy(i,k,1),i=1,imx ),k=1,kmx ),
     >             ((xy(i,k,2),i=1,imx ),k=1,kmx )
        close(1)
        ile = imx/2 + 1
c  fix the leading edge and trainling edge points.
c         ile = 31
        ite = 183
C..read "q" file
       PRINT*,'  Enter Q file name :>'
      READ(*,102) FNAME
      OPEN(2,FILE=FNAME,FORM='FORMATTED',STATUS='OLD')
   10 read(2,*,end=1000) imx,kmx
```

```
        read(2,*) fsmach,alpha,re,time
        read(2,*)
      >    ((( q(i,k,is),i=1,imx ),k=1,kmx ), is=1,4)
C..ROTATE GRID WRT ALPHA  (NOT required for ns.f)
c        CALL ROTXY(ALPHA-alphao,IMX,kMX,XY)
        alphao = alpha
c..extract airfoil coordinates
         ii = 0
         do i = 1,imx
         ii = ii+1
         ax(ii)   = xy(I,1,1)
         ay(ii)   = xy(I,1,2)
         enddo
        PRINT*, ' '
        PRINT*, '  Angle of ATTACK = ', ALPHA
   30   PRINT*, ' '
c        PRINT*, '  Choose the contour function :>'
c        PRINT*, ' '
c        PRINT*, '                         1 : Density'
c        PRINT*, '                         2 : Pressure'
c        PRINT*, '                         3 : Mach Number'
c        PRINT*, '                         4 : Vorticity'
c        PRINT*, '                         5 : Mass-flux'
c        PRINT*, '                         6 : NEXT Time step'
c        PRINT*, '                         7 : EXIT'
c        PRINT*, '                         8 : Next q-file'
c        PRINT*, ' '
c        READ(*,*) IFUN
        do 999 ifun = 1,4
        fmax = -10000.
        fmin = 10000.
        NCON = 25
        IF(IFUN .EQ. 8) THEN
           goto 13
        endif
        IF(IFUN .EQ. 1) THEN
C..ASSIGN DENSITY
         print*,'Density'
        DO 91 k = 1,kMX
        DO 91 I = 1,IMX
        FUNC(I,k) = Q(I,k,1)
        fmin = min(func(i,k), fmin)
        fmax = max(func(i,k), fmax)
   91   continue
        coninc = (fmax-fmin)/ncon
        ELSEIF(IFUN .EQ. 2) THEN
C..EVALUATE Pressure
        print*,'Pressure'
        DO 92 k = 1,kMX
        DO 92 I = 1,IMX
        VEL2      = ( q(i,k,2)**2 + q(i,k,3)**2 )/Q(I,k,1)
        FUNC(I,k) = gam*(gam-1.)*( Q(I,k,4)-.5*VEL2 )
        fmin = min(func(i,k), fmin)
        fmax = max(func(i,k), fmax)
   92   continue
        coninc = (fmax-fmin)/ncon
        ELSEIF(IFUN .EQ. 3) THEN
C..EVALUATE MAch NUMBER
        print*,'Mach'
        DO 93 k = 1,kMX
        DO 93 I = 1,IMX
        VEL2      = ( q(i,k,2)**2 + q(i,k,3)**2 )/Q(I,k,1)**2
           PP        = (GAM-1.)*( Q(I,k,4)-.5*VEL2*Q(I,k,1) )
        AL2       = GAM*PP/Q(I,k,1)
        Func(i,k) = SQRT(VEL2/AL2)
        fmin = min(func(i,k), fmin)
        fmax = max(func(i,k), fmax)
   93   continue
        coninc = (fmax-fmin)/ncon
        ELSEIF(IFUN .EQ. 4) THEN
C..EVALUATE VORTICITY FIELD
        print*,'Vorticity'
        do 41 k=2,kmx-1
          km=k-1
          kp=k+1
        do 41 i=2,imx-1
          ip=i+1
          im=i-1
```

```
            xxi = 0.5 * ( xy(ip,k,1) - xy(im,k,1) )
            zxi = 0.5 * ( xy(ip,k,2) - xy(im,k,2) )
            xze = 0.5 * ( xy(i,kp,1) - xy(i,km,1) )
            zze = 0.5 * ( xy(i,kp,2) - xy(i,km,2) )
            rjacob = 1./( xxi*zze - xze*zxi )
            xix = rjacob * zze
            xiz =-rjacob * xze
            zex =-rjacob * zxi
            zez = rjacob * xxi
         u_ze = 0.5*( q(i,kp,2)/q(i,kp,1)-q(i,km,2)/q(i,km,1))
         w_ze = 0.5*( q(i,kp,3)/q(i,kp,1)-q(i,km,3)/q(i,km,1))
         u_xi = 0.5*( q(ip,k,2)/q(ip,k,1)-q(im,k,2)/q(im,k,1) )
         w_xi = 0.5*( q(ip,k,3)/q(ip,k,1)-q(im,k,3)/q(im,k,1) )
         dudz = u_xi*xiz + u_ze*zez
         dwdx = w_xi*xix + w_ze*zex
         func(i,k) = dudz-dwdx
         fmin = min(func(i,k), fmin)
         fmax = max(func(i,k), fmax)
   41 continue
      do 42 i = 1,imx
   42 func(i,1) = func(i,2)
      fmax = fmax/100.
      fmin = fmin/100.
      coninc = (fmax-fmin)/ncon
      ELSEIF(IFUN .EQ. 5) THEN
C..EVALUATE MASS-FLUX FUNCTION
           c1 = 0.5
           k = 1
           func(1,k) = 0.
           DO  I = 1,imx
             FUNC(I,k) = FUNC(I-1,k) +
     >  C1*( (Q(I,k,2) + Q(I-1,k,2))*(XY(I,k,2)-XY(I-1,k,2)) -
     >         (Q(I,k,3) + Q(I-1,k,3))*(XY(I,k,1)-XY(I-1,k,1)) )
           enddo
c           do  i = iteu, imx
c            func(i,k) = func(imx-i+1,k)
c          enddo
           DO  I=1,IMX
           DO  k=2,kMX
           FUNC(I,k) =  FUNC(I,k-1) +
     > C1*( (Q(I,k,2) + Q(I,k-1,2))*(XY(I,k,2) - XY(I,k-1,2)) -
     >        (Q(I,k,3) + Q(I,k-1,3))*(XY(I,k,1) - XY(I,k-1,1)) )
           fmin = min(func(i,k), fmin)
           fmax = max(func(i,k), fmax)
           enddo
           enddo
           do k=1,kmx
           func(imx,k)= func(1,k)
           end do
            coninc = fsmach/50.
          ncon = (fmax-fmin)/coninc
       ELSEIF(IFUN .EQ. 6) THEN
         goto 10
       ELSEIF(IFUN .EQ. 7) THEN
         goto 1000
       ELSE
         PRINT*, '  WRONG Selection.....'
         goto 30
       ENDIF
   31  PRINT*, '  '
       PRINT*, '  Function Min and Max :>  ',fmin,fmax
       PRINT*, '  '
c      PRINT*, '  Enter NCON :>'
c      READ(*,*) NCON
       conmin = fmin
       conmax = fmax
       fmax = -10000.
       fmin = 10000.
       NP = NP+1
       MNP = MOD(NP,4)
       IF(MNP.EQ.1) THEN
          X0 = 2.37
          Y0 = 4.31
          CALL frmwid(0.012)
          CALL crvwid(0.001)
       ELSEIF(MNP.EQ.2) THEN
          X0 = 6.12
          Y0 = 4.31
```

```
      ELSEIF(MNP.EQ.3) THEN
         X0 = 2.37
         Y0 = 1.31
      ELSEIF(MNP.EQ.0) THEN
         X0 = 6.12
         Y0 = 1.31
      ENDIF
      DUY = YLEN*(XUMAX-XUMIN)/XLEN
      YUMAX =  YUMIN + DUY
      YS = YLEN/(YUMAX-YUMIN)
      XS = XLEN/(XUMAX-XUMIN)
      CALL ORIGIN(X0,Y0)
      CALL SETSUB(XLEN,YLEN)
c     CALL XLABEL(' ',1)
      CALL AXES2D(XUMIN,XUMAX-XUMIN,XUMAX, YUMIN,YUMAX-YUMIN,YUMAX)
      CALL FRAME
      CALL MARGIN(0.)
      DO 50 N = 1,NCON+1
      CONV = CONMIN + coninc*(N-1)
      if( ifun .eq. 4 .and. abs(conv) .lt. .50)  goto 50
      CALL CONTOUR(FUNC,XY,IMX,KMX,CONV,
     >             XUMIN - ABS(XUMIN)*0.5 ,XUMAX*1.5,
     >             YUMIN - ABS(YUMIN)*0.5 ,YUMAX*1.5, NCELL)
      if(ncell .ne. 0) then
        fmin = min(fmin,conv)
        fmax = max(fmax,conv)
c       print*, 'n, conv, ncell :', n,conv,ncell
      endif
   50 CONTINUE
c..nomencleature
      CALL HEIGHT(0.08)
      if( ifun .eq. 1) CALL TXTMSG('DENSITY$',100,0.2, YLEN+0.1)
      if( ifun .eq. 2) CALL TXTMSG('PRESSURE$',100,0.2, YLEN+0.1)
      if( ifun .eq. 3) CALL TXTMSG('MACH NUMBER$',100,0.2, YLEN+0.1)
      if( ifun .eq. 4) CALL TXTMSG('VORTICITY$',100,0.2, YLEN+0.1)
      if( ifun .eq. 5) CALL TXTMSG('MASS-FLUX$',100,0.2, YLEN+0.1)
      call defalf('L/CGREEK')
      call txtmsg('a =$',100, xlen-0.75, ylen-0.2)
      call reset('DEFALF')
      call realno (alpha,1, 'ABUT','ABUT')
      call height(0.05)
      if ( ifun .ne. 5 ) then
c         was 0.24
          call txtmsg ('MIN = $',100, 0.05,0.16)
          call realno (fmin, 2, 'ABUT','ABUT')
          call txtmsg ('MAX = $',100, 0.05,0.05)
          call realno (fmax, 2, 'ABUT','ABUT')
          CALL HEIGHT(0.08)
          call txtmsg('M =$',100,xlen-3.1,ylen-.2 )
          call realno (fsmach,2, 'ABUT','ABUT')
          if (jj .eq. 1) then
          CALL HEIGHT(0.08)
          call txtmsg ('Ramp$',100,xlen-2.95,ylen-.35)
          CALL HEIGHT(0.05)
          call txtmsg ('(A=0.005)$',100,xlen-2.97,ylen-.46)
          elseif (jj .eq. 2) then
          CALL HEIGHT(0.08)
          call txtmsg ('Sinusoid$',100,xlen-3.05,ylen-.35)
          CALL HEIGHT(0.05)
          call txtmsg ('   (k=0.05)$',100,xlen-3.05,ylen-.46)
          endif
c         call txtmsg ('TIME = $',100, 0.6,0.15)
c         call realno (TIME, 2, 'ABUT','ABUT')
      end if
C..draw airfoil..
      call curve (ax,ay, ii, 0)
      IF(MNP .EQ. 1) then
          call height(0.08)
c         call txtmsg('Mach =$',100, 0., -4.1 )
c         call realno (fsmach,2, 'ABUT','ABUT')
c         call txtmsg(',  Re =$',100, 'ABUT', 'ABUT' )
c         call realno (re,1, 'ABUT','ABUT')
c         call txtmsg(ititle,100, 0., -4.4 )
         call endsub(0)
      elseif( mnp .eq. 0) then
        call stoplt(0)
      else
        call endsub(0)
```

```
      endif
c     PRINT*,'  Do you want to change the increament.? (n) :>'
c     READ(*,'(a1)') YN
c     IF(YN.EQ.'Y' .OR. YN.EQ.'y') GOTO 31
c      GOTO 30

  999 continue

 1000 IF(MNP .ne. 0) call stoplt(0)
      CALL FINPLT
      CLOSE(2)
      CLOSE(10)
      STOP
      END
c-----------------------------------------------------------------------
      SUBROUTINE CONTOUR (F,XY,IMX,JMX,CONV,
     >                    XMIN,XMAX,YMIN,YMAX,NCELL)
C..FINDS CONTOUR LINES AND PLOTS
      parameter (idim=213 ,jdim=61 )
      DIMENSION F(IDIM,JDIM),XY(IDIM,JDIM,2), X(2),Y(2)
      NCELL = 0
      DO 50 I = 1, IMX-1
      IP = I+1
      DO 50 J = 1, JMX-1
      JP = J+1
      X1 = XY(I,J,1)
      Y1 = XY(I,J,2)
      IF(X1.GT.XMAX.OR.X1.LT.XMIN.OR.Y1.GT.YMAX.OR.Y1.LT.YMIN)
     >  GOTO 50
      X2 = XY(I,JP,1)
      Y2 = XY(I,JP,2)
      X3 = XY(IP,JP,1)
      Y3 = XY(IP,JP,2)
      X4 = XY(IP,J,1)
      Y4 = XY(IP,J,2)
      F1 = F(I,J)
      F2 = F(I,JP)
      F3 = F(IP,JP)
      F4 = F(IP,J)
      NP = 0
      IF((CONV.GT.F1.AND.CONV.LT.F2) .OR.
     >   (CONV.LT.F1.AND.CONV.GT.F2) ) THEN
         NP = NP+1
         X(NP) = X2 - (F2-CONV)*(X2-X1)/(F2-F1)
         Y(NP) = Y2 - (F2-CONV)*(Y2-Y1)/(F2-F1)
      ENDIF
      IF((CONV.GT.F4.AND.CONV.LT.F3) .OR.
     >   (CONV.LT.F4.AND.CONV.GT.F3) ) THEN
         NP = NP+1
         X(NP) = X3 - (F3-CONV)*(X3-X4)/(F3-F4)
         Y(NP) = Y3 - (F3-CONV)*(Y3-Y4)/(F3-F4)
      ENDIF
      IF(NP.EQ.2) THEN
         CALL RELVEC( X(1),Y(1),X(2),Y(2),0)
         NCELL = NCELL+1
      ELSE
         IF((CONV.GT.F2.AND.CONV.LT.F3) .OR.
     >      (CONV.LT.F2.AND.CONV.GT.F3) ) THEN
            NP = NP+1
            X(NP) = X3 - (F3-CONV)*(X3-X2)/(F3-F2)
            Y(NP) = Y3 - (F3-CONV)*(Y3-Y2)/(F3-F2)
            IF(NP.EQ.2) THEN
              CALL RELVEC( X(1),Y(1),X(2),Y(2),0)
              NCELL = NCELL+1
            ENDIF
         ENDIF
         IF(NP.EQ.1) THEN
            NP = NP+1
            X(NP) = X4 - (F4-CONV)*(X4-X1)/(F4-F1)
            Y(NP) = Y4 - (F4-CONV)*(Y4-Y1)/(F4-F1)
            CALL RELVEC( X(1),Y(1),X(2),Y(2),0)
            NCELL = NCELL+1
         ENDIF
      ENDIF
   50 CONTINUE
      RETURN
      END
C-----------------------------------------------------------------------
```

```
      SUBROUTINE ROTXY ( ANGLE,IMX,JMX,XY )
      parameter (idim=213 ,jdim=61 )
      DIMENSION XY(IDIM,JDIM,2)
      ROTANG = ANGLE*3.14159/180.
      CA =  COS(ROTANG)
      SA = -SIN(ROTANG)
      DO 10 I = 1,IMX
      DO 10 J = 1,JMX
         XC = XY(I,J,1)
         YC = XY(I,J,2)
         XY(I,J,1) = XC*CA - YC*SA
   10    XY(I,J,2) = YC*CA + XC*SA
      RETURN
      END
```

C. PROGRAM NS.F SOUCE CODE

```
      program ns_pot2d
c*********************************************************************
c                                   by
c                          John A. Ekaterinaris
c                      nasa, ames research center
c                              march, 1990
c                              modified by
c                              I.H. Tuncer
c                                  nps
c                              april 1992
c     solution of the 2-D unsteady, thin-layer navier-stokes
c     equations in a time-accurate manner.
c     characteristics of the code:
c     1) factored, iterative, implicit algorithm
c     2) high-order accurate upwind difference scheme (third order)
c     3) baldwin-lomax turbulence model
c     4) patched and overlaid grids
c     5) the code is almost completely vectorized for the cray-ymp
c*********************************************************************
c.."coms.f"
      parameter (nia = 213,   nka = 61)
      common /alpar /oscil, ramp , redfre, omega,
     >              alfa , alfad, alfamn,  alfamx
      logical       oscil, ramp
      common /gamvl /gamma,           gmm,            gmp,
     >              rgamma,           rgmm,           rgmp,
     >              gmbygp
      common /iksri /imx(2),          kmx(2),
     >              imx1(2),          kmx1(2),
     >              imx2(2),          kmx2(2)
      common /ikwk  /iwks(2),         iwke(2)
      common /infvl /rinf,            uinf,           vinf,
     >              winf,             pinf,           einf,
     >              tinf,             amach,          pratio
      common /load  /cl, cd, cm, clv, cdv, cmv
      common /nparm /niter,           newtit,loop,    iter,itr,
     >              nprint,iread,     nload,          odalfa,oalfa
      common /tmval /timeacc, time,   dtau,dt(nia,nka),cour
      common /visci /vismu(nia,nka),  turmu(nia,nka)
      common /visvl /reynnu,reynph,   prkin,          prtur
      common /vispar/visc,  turbl
      common /poten / poten, ntpot,mpot, mdf, ksi, kso, ksiso, sodist
      logical       visc,  turbl,  poten, timeacc
      common /grid  / x(nia,nka), z(nia,nka)
      common /flow  / q(4,nia,nka)
      common /dflow / qd(4,nia,nka)
      common /jacob / aja(nia,nka)
      common /metrcs/ xix(nia,nka), xiz(nia,nka), xit(nia,nka),
     >                zex(nia,nka), zez(nia,nka), zet(nia,nka)
      pi = 4.*atan(1.)
      call data
      do 10 itr = 1, niter
         iter = iter + 1
         time = time + dtau
        alfaold = alfa
        if( oscil ) then
          freq  = 2.*redfre*uinf
```

```
            alfa  = alfamn + 0.5*(alfamx-alfamn)*(1.- cos(freq*time))
             omega = freq  * 0.5*(alfamx-alfamn)*sin(freq*time)
            call grmove(alfa-alfaold)
          elseif( ramp ) then
            omega = 2.*redfre*uinf
            alfa  = alfamn + omega*time
            if( alfa .gt. alfamx) then
               alfa = alfamx
               omega = 0.
            endif
            call grmove(alfa-alfaold)
          endif
          alfad   = alfa * (180./pi)
c..update outer bc
c        if (poten ) call nspot
         call step
         if( abs(alfad-oalfa) .gt. 0.999*odalfa ) then
           call qio(10)
           oalfa = alfad
         endif
         if(mod(iter,1000) .eq. 0 .or. itr .eq. niter) call qio(0)
   10 continue
c     if( .not. poten) then
c       open(unit=33,file='pres.d',form='formatted')
c       kso = 47
c       sso = 0.
c       do i = 2, imx(1)
c       ssoo  =  sso + sqrt( (x(i,kso)-x(i-1,kso))**2 +
c    >                      (z(i,kso)-z(i-1,kso))**2 )
c       pres = gmm*( q(4,i,kso)
c    >           - .5*(q(2,i,kso)**2 + q(3,i,kso)**2)/q(1,i,kso) )
c       sso = ssoo
c       write(33,'(2e15.7)') sso, pres
c       enddo
c     endif
      close(8)
      close(9)
      STOP
      END
c     include 'nspot.f'
C-----------------------------------------------------------------------
      subroutine data
      include 'coms.f'
      pi = 4.*atan(1.)
      read (5,*)
      read (5,*)
      read (5,*)
c..read top comments
      read (5,*)
      read (5,*) iread, niter, nprint, nload, odalfa
      read (5,*)
      read (5,*) poten, ntpot, mpot, mdf, ksiso, sodist
      read (5,*)
      read (5,*) alfad, oscil, ramp, redfre, alfamnd, alfamxd
      read (5,*)
      read (5,*) amach, reynph, visc,  turbl
      read (5,*)
      read (5,*) timeacc, cour, newtit
c.. read in grid
      open(unit=11,file='grid.in',form='formatted',status='old')
      nt = 1
      ng = 1
      read (11,*) imx(nt), kmx(nt), iwks(1), iwke(1)
      read (11,*) ((x(i,k), i = 1, imx(1)), k = 1, kmx(1) ),
     >            ((z(i,k), i = 1, imx(1)), k = 1, kmx(1) )
      print *, 'Grid dimensions are :', imx(nt), kmx(nt)
      print *, 'TE is located at I =', iwks(1)
      kmx(2)   = kmx(1)
      imx1(nt) = imx(nt) - 1
      imx2(nt) = imx(nt) - 2
      kmx1(nt) = kmx(nt) - 1
      kmx2(nt) = kmx(nt) - 2
      alfa   = alfad*pi/180.
      alfamn = alfamnd*pi/180.
      alfamx = alfamxd*pi/180.
      omega = 0.
c.. specify some parameters ***
      tinf   = 530.0
```

```
        prkin  = 0.72
        prtur  = 0.90
        reynnu  = reynph / amach
        gamma   = 1.4
        gmm     = gamma - 1.0
        gmp     = gamma + 1.0
        rgamma  = 1.0/gamma
        rgmm    = 1.0/gmm
        rgmp    = 1.0/gmp
        gmbygp  = gmm/gmp
        rinf    = 1.0
        uinf    = amach
        winf    = 0.
        pinf    = 1.0/gamma
        einf    = 0.5*rinf*(uinf**2 + winf**2) + pinf*rgmm
        iter = 0
        time = 0.
        if (iread .ne. 0) then
          call qio(iread)
           kmx1(1) = kmx(1) - 1
           kmx2(1) = kmx(1) - 2
           alfa = alfad*pi/180.
           call grmove(alfa)
            if( (oscil .or. ramp) .and.
     >              abs(alfad - alfamnd) .lt. 1.e-05 ) then
              iter = 0
              time = 0.
            endif
c       elseif( poten ) then
c..initialize bl (k<kpots) and the potential flowfield..
c          kpots = 25
c            do i = 1,imx(1)
c            do k = 1,kmx(1)
c            q(1,i,k) = rinf
c   enddo
c            q(2,i,1) = 0.
c   if (i .lt. iwks(1) .or. i .gt. iwke(1)) q(2,i,1) = uinf
c            q(3,i,1) = 0.
c            q(4,i,1) = einf
cenddo
c          call grmove(alfa)
c          call potsv(1,kmx(1), kpots, iwks(1), uinf)
c          do k = 2, kpots-1
c          ratiok = float(k)/kpots
c          do l = 1,4
c          do i = 1,imx(1)
c            if (i .ge. iwks(1) .and. i .le. iwke(1)) then
c             q(l,i,k) = q(l,i,1) + ratiok*( q(l,i,kpots)-q(l,i,1) )
c            else
c             q(l,i,k) = q(l,i,kpots)
c            endif
c          enddo
c          enddo
c          enddo
        else
c.. initialize q to freestream values everywhere ***
          do 60 k = 1, kmx(1)
          fact = min(1., float(k)/15)
          do 60 i = 1, imx(1)
              q(1,i,k) = rinf
            if (i .ge. iwks(1) .and. i .le. iwke(1)) then
              q(2,i,k) = fact * rinf*uinf
            else
              q(2,i,k) = rinf*uinf
            endif
              q(3,i,k) = 0.
              q(4,i,k) = einf
 60        continue
           call grmove(alfa)
        endif
        call eigen
        oalfa = float(int(alfad))
        open(unit=9,file='loads.d',form='formatted')
        open(unit=8,file='qp.d',form='unformatted')
        write(6,101)
  101 format(//' Iter Alpha   Time    Resid      Density  i   k    ',
     >'    Cm      Cd       Cl')
        return
```

```
      end
C------------------------------------------------------------------
      subroutine bc
      include 'coms.f'
      logical doit
c     if ( poten .or. ramp .or. oscil .or. iter .gt. 50) then
         decay = 0.
c     else
c        decay  = 1.0 - float(iter)/50.
c     endif
      ng    = 1
c..for <k=1>, <i=itel,iteu> for airfoils ***
      i1 = iwks(ng)
      i2 = iwke(ng)
      k1 = 1
      k2 = 2
      k3 = 3
      do 100 i=i1,i2
          rval2 = q(1,i,k2)
          pval2 = gmm*(q(4,i,k2) -
     >            0.5*( q(2,i,k2)**2 + q(3,i,k2)**2 ) / q(1,i,k2))
          rval1 = rval2
          pval1 = pval2
          xtau  = omega * z(i,k1)
          ztau  =-omega * x(i,k1)
          if( visc )  then
c..enforce non-slip boundary condition on the surface ***
             u1   = q(2,i,k1)
             u2   = q(3,i,k1)
          else
c..enforce slip boundary condition on the surface ***
             decay = 1.
             u2      = q(2,i,k2)/q(1,i,k2)
             u3      = q(2,i,k3)/q(1,i,k3)
             v2      = q(3,i,k2)/q(1,i,k2)
             v3      = q(3,i,k3)/q(1,i,k3)
             ucon2   = xtau + xix(i,k2)*u2 + xiz(i,k2)*v2
             ucon3   = xtau + xix(i,k3)*u3 + xiz(i,k3)*v3
             vcon2   = ztau + zex(i,k2)*u2 + zez(i,k2)*v2
             vcon3   = ztau + zex(i,k3)*u3 + zez(i,k3)*v3
             ucon1   = 2.*ucon2 - ucon3
             vcon1   = 0.
             u1      = ( (ucon1-xtau)*zez(i,k1) + xiz(i,k1)*ztau )
     >                           *aja(i,k1)
             v1      = (-(ucon1-xtau)*zex(i,k1) - xix(i,k1)*ztau )
     >                           *aja(i,k1)
          endif
          q(1,i,k1) = rval1
          q(2,i,k1) = (decay * u1 + xtau) * rval1
          q(3,i,k1) = (decay * v1 + ztau) * rval1
          q(4,i,k1) = rgmm*pval1 + 0.5*(q(2,i,k1)**2+q(3,i,k1)**2 )
     >                                       / rval1
 100   continue
       doit = .false.
       if( .not. poten .and. doit ) then
c..set free-stream conditions..
          q(1,i,k1) = rinf
          q(2,i,k1) = rinf*uinf
          q(3,i,k1) = 0.
          q(4,i,k1) = einf
       elseif( .not. poten ) then
c.. enforce boundary conditions at the inlet boundary
c         1) pt(kmax) = pt(inlet) (total pressure condition)
c         2)  u(kmax) = uinf    (inlet angle = a degs)
c         3)  w(kmax) = winf    (inlet angle = a degs)
c         4)  reim1(kmax) = reim1(inlet)
c                      (reimann variable reim1 = u + 2c/(gamma-1))
c         5)  reim2(kmax) = reim2(kmax-1)
c                      (reimann variable reim2 = u - 2c/(gamma-1)) ***
c
         cinf  = sqrt(gamma*pinf/rinf)
         sinf  = pinf/rinf**gamma
         reim1 = uinf + 2.0*cinf*rgmm
         ptinf = pinf*(1.0 + 0.5*gmm*amach**2)**(gamma*rgmm)
         k1    = kmx(ng)
         k2    = kmx1(ng)
         do 200 i = 2, imx1(ng)
          pval2 = gmm*(q(4,i,k2) - 0.5*( q(2,i,k2)**2 + q(3,i,k2)**2 )
```

```
     >                                                  / q(1,i,k2) )
          cval2 = sqrt(gamma*pval2/q(1,i,k2))
          uval2 = q(2,i,k2)/q(1,i,k2)
          reim2 = uval2 - 2.0*cval2*rgmm
          uval1 = 0.5*(reim1 + reim2)
          wval1 = winf
          cval1 = 0.25*gmm*(reim1 - reim2)
c         amsq  = (uval1**2 + wval1**2)/cval1**2
c         pval1 = ptinf*(1.0 + 0.5*gmm*amsq)**(-gamma*rgmm)
c         rval1 = gamma*pval1/cval1**2
          sval1 = sinf
          rval1 = (rgamma*cval1**2/sval1)**rgmm
          pval1 = rgamma*rval1*cval1**2
          q(1,i,k1) = rval1
          q(2,i,k1) = rval1*uval1
          q(3,i,k1) = rval1*wval1
          q(4,i,k1) = pval1*rgmm + 0.5*rval1*(uval1**2 +wval1**2)
  200   continue
       endif
c..enforce boundary conditions at the exit boundary
c         1) p = pstat (static pressure condition)
c         2) w(imax)=  w(imax-1)
c         2) reiml(imax) = reiml(imax-1)
c                     (reimann variable reiml= u + 2c/(gamma-1))
c         4) s(imax) = s(imax-1) (entropy condition) ***
c
      ng    = 1
      i1 = imx(ng)
      i2 = imx1(ng)
      do 300 k = 1, kmx(ng)
          rval2 = q(1,i2,k)
          pval2 = gmm*(q(4,i2,k) - 0.5*( q(2,i2,k)**2 + q(3,i2,k)**2 )
     >                                               / q(1,i2,k) )
          cval2 = sqrt(gamma*pval2/q(1,i2,k))
          uval2 = q(2,i2,k)/q(1,i2,k)
          wval2 = q(3,i2,k)/q(1,i2,k)
          reiml = uval2 + 2.0*rgmm*cval2
          sval2 = pval2/rval2**gamma
          sval1 = sval2
          pval1 = pinf
          rval1 = (pval1/sval1)**rgamma
          cval1 = sqrt(gamma*pval1/rval1)
          uval1 = min( uinf, reiml - 2.0*rgmm*cval1 )
          wval1 = wval2
c                       rval1 = rval2
c                       uval1 = uval2
c                       wval1 = wval2
          q(1,i1,k) = rval1
          q(2,i1,k) = rval1*uval1
          q(3,i1,k) = rval1*wval1
          q(4,i1,k) = pval1*rgmm + 0.5*rval1*(uval1**2 + wval1**2)
 300  continue
      i1 = 1
      i2 = 2
      do 350 k = 1, kmx(ng)
          rval2 = q(1,i2,k)
          pval2 = gmm*(q(4,i2,k) - 0.5*( q(2,i2,k)**2 + q(3,i2,k)**2 )
     >                                                / q(1,i2,k))
          cval2 = sqrt(gamma*pval2/q(1,i2,k))
          uval2 = q(2,i2,k)/q(1,i2,k)
          wval2 = q(3,i2,k)/q(1,i2,k)
          reiml = uval2 + 2.0*rgmm*cval2
          sval2 = pval2/rval2**gamma
          sval1 = sval2
          pval1 = pinf
          rval1 = (pval1/sval1)**rgamma
          cval1 = sqrt(gamma*pval1/rval1)
          uval1 = reiml - 2.0*rgmm*cval1
          wval1 = wval2
          q(1,i1,k) = rval1
          q(2,i1,k) = rval1*uval1
          q(3,i1,k) = rval1*wval1
          q(4,i1,k) = pval1*rgmm + 0.5*rval1*(uval1**2 + wval1**2)
 350  continue
c..outgoing bc along the C part of the grid..
      doit = .false.
      if ( doit) then
        k1 = kmx(1)
```

```
          k2 = kmx1(1)
          do 360 i = 2, imx1(1)
          vxs = (z(i,k1)-z(i-1,k1))*q(2,i,k1) -
     >         (x(i,k1)-x(i-1,k1))*q(3,i,k1)
          if( vxs .lt. 0. ) then
c           print*, 'Outgoing bc at i:', i, pval1
c           rval2 = q(1,i,k2)
c           pval2 = gmm*(q(4,i,k2) - 0.5*( q(2,i,k2)**2 + q(3,i,k2)**2 )
c    >                                               / q(1,i,k2))
c           cval2 = sqrt(gamma*pval2/rval2
c           uval2 = q(2,i,k2)/q(1,i,k2)
c           wval2 = q(3,i,k2)/q(1,i,k2)
c           reim1 = uval2 + 2.0*rgmm*cval2
c           sval2 = pval2/rval2**gamma
c           sval1 = sval2
c           pval1 = pinf
c           if (poten) pval1 = gmm*(q(4,i,k1) -
c    >                 0.5*( q(2,i,k1)**2 + q(3,i,k1)**2 ) / q(1,i,k1))
c           rval1 = (pval1/sval1)**rgamma
c           cval1 = sqrt(gamma*pval1/rval1)
c           uval1 = reim1 - 2.0*rgmm*cval1
c           wval1 = wval2
c           q(1,i,k1) = rval1
c           q(2,i,k1) = rval1*uval1
c           q(3,i,k1) = rval1*wval1
c           q(4,i,k1) = pval1*rgmm + 0.5*rval1*(uval1**2 + wval1**2)
            pval1 = gmm*(q(4,i,k1) -
     >                 0.5*( q(2,i,k1)**2 + q(3,i,k1)**2 ) / q(1,i,k1))
c           q(1,i,k1) = (4.*q(1,i,k2) - q(1,i,k2-1))/3.
            q(1,i,k1) = 2.*q(1,i,k2) - q(1,i,k2-1)
            q(2,i,k1) = 2.*q(2,i,k2) - q(2,i,k2-1)
            q(3,i,k1) = 2.*q(3,i,k2) - q(3,i,k2-1)
            q(4,i,k1) = pval1*rgmm +
     >                 0.5*( q(2,i,k1)**2 + q(3,i,k1)**2 ) / q(1,i,k1)
c         else
c           q(1,i,k1) = 2.*q(1,i,k2) - q(1,i,k2-1)
          endif
360     continue
        endif
c..boundary condition treatment for the wake ***
        ng = 1
        k = 1
        i1 = iwks(ng)-1
        do 400 l = 1,4
        do 400 i = 1,i1
        il = i
        iu = imx(ng) - i + 1
c..average values on upper and lower surfaces of cut,
        q(l,il,k) = 0.5*( q(l,iu,k+1) + q(l,il,k+1) )
        q(l,iu,k) = 0.5*( q(l,iu,k+1) + q(l,il,k+1) )
400     continue
        return
        end
C-----------------------------------------------------------------
        subroutine step
        include 'coms.f'
        dimension qold(4,nia,nka)
        nt = 1
c.. store all the q values to facilitate an iterative update ***
        do 1 l=1,4
        do 1 i=1,imx(1)
          do 1 k=1,kmx(1)
          qold(l,i,k) = q(l,i,k)
1       continue
c.. update all the q values ***
        DO 1000 loop = 1, newtit
c..update outer bc
c       if (poten .and. loop .eq. 1 ) call nspot
c       if (.not. poten .and. loop .eq. 1 ) then
c..write out dphi/dt related terms..
c           rho  =  q(1,61,39)
c           pres = gmm*(q(4,61,39) - 0.5*( q(2,61,39)**2 + q(3,61,39)**2 )
c    >                                                   / rho )
c           v2   =   (q(2,61,39)/rho)**2 + (q(3,61,39)/rho)**2
c           dfdt =  0.5*(uinf**2-v2) + rgmm*(1.-(gamma*pres)**(gmm/gamma))
c write(6,'(3x,e14.4,14x,3e14.4)') dfdt, rho, pres, v2
c       endif
c..update all the qsi values
```

```
         do 10 k = 1, kmx(nt)
           do 10 l = 1, 4
             do 10 i = 1, imx(nt)
             qd(l,i,k) = -( q(l,i,k) - qold(l,i,k) )/aja(i,k)
 10      continue
         call rhsosh
         call lhs
         adromax = 0.0
         do 20 k = 2, kmxl(nt)
         do 20 i = 2, imxl(nt)
             dro    = aja(i,k)*qd(1,i,k)
             adro   = abs(dro)
             if ( adromax .lt. adro) then
               dromax = dro
               adromax = adro
               ires = i
               kres = k
             endif
             if(.not.(adro .lt. 5.0 .and. adro .ge. 0.))then
              write(6,101) iter, adro, i,k
   101         format(//'  *** It has BLOWN UP ***  @ ITER =', i5 /
     >          '            drho = ',e11.3, ' @ ', 2i4 )
              stop
             endif
 20      continue

         do 30 k = 2, kmxl(nt)
           do 30 l = 1, 4
             do 30 i = 2, imxl(nt)
               q(l,i,k) = q(l,i,k) + aja(i,k)*qd(l,i,k)
 30      continue
         call bc
 1000  CONTINUE
       if( mod(iter,nload).eq.0 .or.
     >       itr .eq. 1 .or. itr .eq. niter) then
         call loads
         write (6,60) iter,alfad,time,dromax,
     >               q(1,ires,kres), ires,kres,cm,cd,cl
   60    format(i5 ,f6.2, 1x,f8.4, e11.3,f9.4, 2i4, 2x,3f8.4)
************* Cl loop
c      if ((.not.(oscil.or.ramp)) .and.(abs(clo-cl).lt..001))then
c         call qio(0)
c         stop
c      endif
c      clo=cl
*************
       elseif( mod(iter,nprint).eq.0) then
        write (6,60) iter,alfad,time,dromax,
     >               q(1,ires,kres), ires,kres
       endif
       return
       end
C------------------------------------------------------------------------
       subroutine rhsosh
       include 'coms.f'
       common /dfdq  / ap(nia,4,4), am(nia,4,4)
       common /fmet  /aktj(nia),aktnj(nia),
     >               akxj(nia),akzj(nia),ajac(nia,2),adt(nia)
       common /flux  /qv(nia,2,4),fnum(nia,4),dfp(nia,4,3),dfm(nia,4,3)
     >                                       ,dvq(nia,4,3)
       common /osflxs/ dfpt(nia,4),dfmt(nia,4)
       common /osvars/ r(nia,6),u(nia,6),v(nia,6),e(nia,6)
       dimension akx(nia), akz(nia), akt(nia), aktn(nia)
       nt=1
       oneby6=1.0/6.0
c
       do 1000 k=2,kmxl(nt)
c**********************************************************************
c      compute the fluxes for all the segments in the psi direction
c**********************************************************************
       do 110 i=2,imx(nt)
       i1=i-1
       i4=i
       r(i,1)=q(1,i1,k)
       u(i,1)=q(2,i1,k)/r(i,1)
       v(i,1)=q(3,i1,k)/r(i,1)
       e(i,1)=q(4,i1,k)
       r(i,4)=q(1,i4,k)
```

```
      u(i,4)=q(2,i4,k)/r(i,4)
      v(i,4)=q(3,i4,k)/r(i,4)
      e(i,4)=q(4,i4,k)
      xi_x=0.5*(xix(i1,k)+xix(i4,k))
      xi_z=0.5*(xiz(i1,k)+xiz(i4,k))
      xi_t=0.5*(xit(i1,k)+xit(i4,k))
      ze_t=0.5*(zet(i1,k)+zet(i4,k))
      aktj(i) = xi_t
      aktnj(i)= ze_t
      akxj(i) = xi_x
      akzj(i) = xi_z
110   continue
      ilft=2
      irgt=imx(nt)
      call osflux(ilft,irgt)
c**********************************************************************
c     add the fluxes in each subpath
c**********************************************************************
      do 120 n=1,4
      do 120 i=ilft,irgt
      dfpt(i,n)=dfp(i,n,1)+dfp(i,n,2)+dfp(i,n,3)
      dfmt(i,n)=dfm(i,n,1)+dfm(i,n,2)+dfm(i,n,3)
120   continue
c*********************************************************************
c     add the eta flux contribution for the second and last but one
c     points (second order accurate fluxes)
c*********************************************************************
      idif=imx(nt)-3
      do 130 n=1,4
      do 130 i=2,imx1(nt),idif
      ip0=i
      ip1=i+1
      qd(n,i,k)=qd(n,i,k)-dt(i,k)*(fnum(ip1,n)-fnum(ip0,n)+
     > 0.45*(dfpt(ip1,n)-dfpt(ip0,n)-dfmt(ip1,n)+dfmt(ip0,n)))
130   continue
c*********************************************************************
c     add the eta flux contribution for the points in the interior
c     points (third order accurate fluxes)
c*********************************************************************
      do 140 n=1,4
      do 140 i=3,imx2(nt)
      im1=i-1
      ip0=i
      ip1=i+1
      ip2=i+2
      qd(n,i,k)=qd(n,i,k)-dt(i,k)*(fnum(ip1,n)-fnum(ip0,n)
     >             +oneby6*(2.0*dfpt(ip1,n)-dfpt(ip0,n)-dfpt(im1,n))
     >             +oneby6*(2.0*dfmt(ip0,n)-dfmt(ip1,n)-dfmt(ip2,n)))
140   continue
1000  continue
      do 2000 i=2,imx1(nt)
c**********************************************************************
c     compute the fluxes for all the segments in the zet direction
c**********************************************************************
      if( i .lt. iwks(nt) .or. i .gt. iwke(nt) ) then
        kbot = 1
      else
        kbot = 2
      endif
      ktop=kmx(nt)
      do 210 k=kbot,kmx(nt)
      if(k .eq. 1) then
        ii  = imx(nt)-i+1
        k1 = 2
        sign=-1.
      else
        ii = i
        k1 = k-1
        sign=1.
      endif
      k4=k
      r(k,1)=q(1,ii,k1)
      u(k,1)=q(2,ii,k1)/r(k,1)
      v(k,1)=q(3,ii,k1)/r(k,1)
      e(k,1)=q(4,ii,k1)
      r(k,4)=q(1,i,k4)
      u(k,4)=q(2,i,k4)/r(k,4)
      v(k,4)=q(3,i,k4)/r(k,4)
```

```
      e(k,4)=q(4,i,k4)
      ze_x=0.5*(sign*zex(ii,k1)+zex(i,k4))
      ze_z=0.5*(sign*zez(ii,k1)+zez(i,k4))
      ze_t=0.5*(sign*zet(ii,k1)+zet(i,k4))
      xi_t=0.5*(sign*xit(ii,k1)+xit(i,k4))
      aktnj(k)=xi_t
      aktj(k)=ze_t
      akxj(k)=ze_x
      akzj(k)=ze_z
210   continue
      call osflux(kbot,ktop)
c***********************************************************************
c     add the fluxes in each subpath
c***********************************************************************
      do 220 n=1,4
      do 220 k=kbot,ktop
      dfpt(k,n)=dfp(k,n,1)+dfp(k,n,2)+dfp(k,n,3)
      dfmt(k,n)=dfm(k,n,1)+dfm(k,n,2)+dfm(k,n,3)
220   continue
c**********************************************************************
c     add the eta flux contribution for the last
c     point (first, or second order accurate fluxes)
c**********************************************************************
c  -- second order at the inner boundaries --
c     qd(n,i,k) = qd(n,i,k)-dt(i,k)*(fnum(kp1,n)-fnum(kp0,n)+
c    > 0.45*(dfpt(kp1,n)-dfpt(kp0,n)-dfmt(kp1,n)+dfmt(kp0,n)))
      do n = 1,4
      if(kbot .eq. 2 ) then
       k=2
       kp0=k
       kp1=k+1
c      qd(n,i,k) = qd(n,i,k)-dt(i,k)*(fnum(kp1,n)-fnum(kp0,n)+
c    >  0.45*(dfpt(kp1,n)-dfpt(kp0,n)-dfmt(kp1,n)+dfmt(kp0,n)))
       qd(n,i,k) = qd(n,i,k)-dt(i,k)*( fnum(kp1,n)-fnum(kp0,n) )
      endif
       k=kmx1(nt)
       kp0=k
       kp1=k+1
       qd(n,i,k) = qd(n,i,k)-dt(i,k)*( fnum(kp1,n)-fnum(kp0,n) )
       k=kmx2(nt)
       kp0=k
       kp1=k+1
       qd(n,i,k) = qd(n,i,k)-dt(i,k)*( fnum(kp1,n)-fnum(kp0,n) )
      enddo
c**********************************************************************
c     add the eta flux contribution for the points in the interior
c     points (third order accurate fluxes)
c**********************************************************************
      do 240 n=1,4
      do 240 k=kbot+1,kmx2(nt)-1
      km1=k-1
      kp0=k
      kp1=k+1
      kp2=k+2
      qd(n,i,k)=qd(n,i,k)-dt(i,k)*(fnum(kp1,n)-fnum(kp0,n)
     >            +oneby6*(2.0*dfpt(kp1,n)-dfpt(kp0,n)-dfpt(km1,n))
     >            +oneby6*(2.0*dfmt(kp0,n)-dfmt(kp1,n)-dfmt(kp2,n)))
240   continue
2000  continue
      if( visc ) call oshvrhs
      return
      end
C----------------------------------------------------------------------
      subroutine lhs
      parameter (nikp = 213,   ninv=61)
      include 'coms.f'
      common /ctri  /amat(ninv,nikp,4,4), bmat(ninv,nikp,4,4),
     >               cmat(ninv,nikp,4,4), fmat(ninv,nikp,4)
      common /swvar / rsw(nia),usw(nia),vsw(nia),esw(nia)
      common /dfdq  / ap(nia,4,4), am(nia,4,4)
      common /fmet  /aktj(nia),aktnj(nia),
     >               akxj(nia),akzj(nia),ajac(nia,2),adt(nia)
      common /flux  /qv(nia,2,4),fnum(nia,4),dfp(nia,4,3),dfm(nia,4,3)
     >                                      ,dvq(nia,4,3)
      common /visdi /vmui(nka),      tmui(nka)
      dimension      iline(nikp),      kline(nikp)
      nt=1
      ng = 1
```

```
c..inversion in the psi direction ***
      klft=2
      krgt=kmx1(nt)
      line=0
      do 1000 k=klft,krgt
        kp1=k+1
        km1=k-1
          line=line+1
          if(line.gt.ninv) line=line-ninv
          kline(line)=k
c..initialize the matrices ***
          do 110 l=1,4
            do 110 i=1,imx(nt)
              fmat(line,i,l)=qd(l,i,k)
110          continue
c..calculate the matrices aplus and aminus ***
          do 120 i=1,imx(nt)
            yac     = aja(i,k)
            akxj(i)  = xix(i,k) * yac
            akzj(i)  = xiz(i,k) * yac
            aktj(i)  = xit(i,k) * yac
           adt(i)    = dt(i,k)
            ajac(i,1)= aja(i,k)
            ajac(i,2)= aja(i,k)
             rsw(i)  = q(1,i,k)
             usw(i)  = q(2,i,k)/q(1,i,k)
             vsw(i)  = q(3,i,k)/q(1,i,k)
             esw(i)  = q(4,i,k)
120          continue
          do 130 l=1,4
            do 130 i=1,imx(nt)
              qv(i,1,l)=q(l,i,k)
              qv(i,2,l)=q(l,i,k)
130       continue
          ilft=1
          irgt=imx(nt)
          call smatrx(ilft,irgt)
c.. calculate the matrices amat, bmat and cmat ***
          do 140 l=1,4
            do 140 m=1,4
              do 140 i=2,imx1(nt)
              ip = i+1
              im = i-1
              amat(line,i,l,m)=-ap(im,l,m)
              bmat(line,i,l,m)= ap(i ,l,m)-am(i,l,m)
              cmat(line,i,l,m)= am(ip,l,m)
140       continue

c..add the contribution from the time term ***
          ilft=1
          irgt=imx(nt)
          do 150 l=1,4
            do 150 i=ilft,irgt
              bmat(line,i,l,l)=bmat(line,i,l,l)+1.0
150       continue
            i=1
            do 160 l=1,4
                fmat(line,i,l)=0.0
              do 160 m=1,4
                amat(line,i,l,m)=0.0
                bmat(line,i,l,m)=0.0
                cmat(line,i,l,m)=0.0
160       continue
              do 161 l=1,4
                bmat(line,i,l,l) = 1
161       continue
            i=imx(nt)
            do 165 l=1,4
                fmat(line,i,l)=0.0
              do 165 m=1,4
                amat(line,i,l,m)=0.0
                bmat(line,i,l,m)=0.0
                cmat(line,i,l,m)=0.0
165       continue
              do 166 l=1,4
                bmat(line,i,l,l) = 1
166       continue
          if( line .eq. ninv .or. k. eq. krgt ) then
```

```
c..solve the block tridiagonal system ***
            call btri(1,imx(nt),line)
c..redefine the rhs vector ***
            do 170 lcount=1,line
              kd=kline(lcount)
              do 170 l=1,4
              do 170 id=1,imx(nt)
                qd(l,id,kd)=fmat(lcount,id,l)
170         continue
          endif
1000    continue
c..inversion in the zeta direction ***
      ilft=2
      irgt=imxl(nt)
      line=0
        do 2000 i=ilft,irgt
          ipl=i+1
          iml=i-1
          line=line+1
          if(line.gt.ninv) line=line-ninv
          iline(line)=i
c..initialize the matrices ***
          do 210 l=1,4
            do 210 k=1,kmx(nt)
              fmat(line,k,l)=qd(l,i,k)
210       continue
c..calculate the matrices aplus and aminus ***
          do 220 k=1,kmx(nt)
            yac       = aja(i,k)
            akxj(k)   = zex(i,k) * yac
            akzj(k)   = zez(i,k) * yac
            aktj(k)   = zet(i,k) * yac
            adt(k)    = dt(i,k)
            ajac(k,1)= aja(i,k)
            ajac(k,2)= aja(i,k)
             rsw(k)   = q(1,i,k)
             usw(k)   = q(2,i,k)/q(1,i,k)
             vsw(k)   = q(3,i,k)/q(1,i,k)
             esw(k)   = q(4,i,k)
220       continue
          do 230 l=1,4
            do 230 k=1,kmx(nt)
              qv(k,1,l)=q(l,i,k)
              qv(k,2,l)=q(l,i,k)
230       continue
          klft=1
          krgt=kmx(nt)
          call smatrx(klft,krgt)

c..calculate the matrices amat, bmat and cmat ***
          do 240 l=1,4
            do 240 m=1,4
              do 240 k=2,kmxl(nt)
                kp=k+1
                km=k-1
                amat(line,k,l,m)=-ap(km,l,m)
                bmat(line,k,l,m)= ap(k ,l,m)-am(k,l,m)
                cmat(line,k,l,m)= am(kp,l,m)
240       continue
c..add the viscous eta contribution to the lhs ***
          if ( visc ) then
          klft=1
          krgt=kmx(nt)
          do 300 k=klft,krgt
            vmui(k)=vismu(i,k)
            tmui(k)=turmu(i,k)
            akxj(k)   = zex(i,k)
            akzj(k)   = zez(i,k)
            aktj(k)   = zet(i,k)
300       continue
          call vmatrx(klft,krgt)
          do 310 l=1,4
            do 310 m=1,4
              do 310 k=2,kmxl(nt)
                kp=k+1
                km=k-1
                amat(line,k,l,m)=amat(line,k,l,m)-ap(km,l,m)
                bmat(line,k,l,m)=bmat(line,k,l,m)+ap(k ,l,m)+
```

```
     >                                                    ap(k ,l,m)
                 cmat(line,k,l,m)=cmat(line,k,l,m)-ap(kp,l,m)
 310        continue
            endif
c..add the contribution from the time term ***
            klft=1
            krgt=kmx(nt)
            do 250 l=1,4
              do 250 k=klft,krgt
                bmat(line,k,l,l)=bmat(line,k,l,l)+1.0
 250        continue
            k=1
            do 330 l=1,4
              fmat(line,k,l)=0.0
              do 330 m=1,4
                amat(line,k,l,m)=0.0
                bmat(line,k,l,m)=0.0
                cmat(line,k,l,m)=0.0
 330        continue
              do 331 m=1,4
                bmat(line,k,m,m)=1.0
 331        continue
            k=kmx(nt)
            do 340 l=1,4
              fmat(line,k,l)=0.0
              do 340 m=1,4
                amat(line,k,l,m)=0.0
                bmat(line,k,l,m)=0.0
                cmat(line,k,l,m)=0.0
 340        continue
              do 341 m=1,4
                bmat(line,k,m,m)=1.0
 341        continue
            if( line .eq. ninv .or. i.eq.irgt ) then
c..solve the block tridiagonal system ***
              call btri(1,kmx(nt),line)
c..redefine the rhs vector ***
              do 270 lcount=1,line
                id=iline(lcount)
                do 270 l=1,4
                do 270 kd=1,kmx(nt)
                  qd(l,id,kd)=fmat(lcount,kd,l)
 270          continue
            endif
 2000    continue
        return
        end

c-------------------------------------------------------------------
        subroutine osflux(lbeg,lend)
        parameter (nia = 213,   nka = 61)
        common /gamvl /gamma,            gmm,                gmp,
     >               rgamma,             rgmm,               rgmp,
     >               gmbygp
        common /fmet  /aktj(nia),aktnj(nia),
     >               akxj(nia),akzj(nia),ajac(nia,2),adt(nia)
        common /flux  /qv(nia,2,4),fnum(nia,4),dfp(nia,4,3),dfm(nia,4,3)
     >                                    ,dvq(nia,4,3)
        common /osflxs/ dfpt(nia,4),dfmt(nia,4)
        common /osvars/ r(nia,6),u(nia,6),v(nia,6),e(nia,6)
        dimension eig11(nia),eig12(nia),eig21(nia),eig22(nia),eig31(nia)
        dimension eig32(nia),p(nia,6),c(nia,6),fact(nia)
        dimension fvs(nia,4,6),u_b(6),v_b(6),vqs(nia,4,4)
c*******************************************************************
c       define constants for the osher scheme
c*******************************************************************
        osher=-1.0
        exp=0.5*gmm
        rexp=1.0/exp
c*******************************************************************
c       calculate the intermediate quantities
c*******************************************************************
        do 10 l=lbeg,lend
        p(l,1)=gmm*(e(l,1)-0.5*r(l,1)*(u(l,1)**2+v(l,1)**2))
        c(l,1)=sqrt(gamma*p(l,1)/r(l,1))
        p(l,4)=gmm*(e(l,4)-0.5*r(l,4)*(u(l,4)**2+v(l,4)**2))
        c(l,4)=sqrt(gamma*p(l,4)/r(l,4))
        fact(l)=sqrt(akxj(l)**2+akzj(l)**2)
```

```
      ofact   =1.0/fact(l)
      aktj(l) =ofact*aktj(l)
      akxj(l) =ofact*akxj(l)
      akzj(l) =ofact*akzj(l)
      aktnj(l)=ofact*aktnj(l)
 10   continue
      do 20 l=lbeg,lend
      u_b(1)=u(l,1)*akxj(l)+v(l,1)*akzj(l)
      v_b(1)=v(l,1)*akxj(l)-u(l,1)*akzj(l)
      u_b(4)=u(l,4)*akxj(l)+v(l,4)*akzj(l)
      v_b(4)=v(l,4)*akxj(l)-u(l,4)*akzj(l)
      az=(p(l,4)/p(l,1))**(0.5*rgamma)/sqrt(r(l,4)/r(l,1))
      u_b(2)=(u_b(4)+osher*rexp*c(l,4)+az*(u_b(1)
     >                -osher*rexp*c(l,1)))/(1.0+az)
      u_b(3)=u_b(2)
      v_b(2)=v_b(1)
      v_b(3)=v_b(4)
      c(l,2)=c(l,1)+osher*exp*(u_b(2)-u_b(1))
      c(l,3)=c(l,4)+osher*exp*(u_b(4)-u_b(3))
      r(l,2)=r(l,1)*(c(l,2)/c(l,1))**rexp
      r(l,3)=r(l,4)*(c(l,3)/c(l,4))**rexp
      p(l,2)=rgamma*r(l,2)*c(l,2)**2
      p(l,3)=p(l,2)

      u(l,2)=akxj(l)*u_b(2)-akzj(l)*v_b(2)
      v(l,2)=akzj(l)*u_b(2)+akxj(l)*v_b(2)
      e(l,2)=0.5*r(l,2)*(u(l,2)**2+v(l,2)**2)+rgmm*p(l,2)

      u(l,3)=akxj(l)*u_b(3)-akzj(l)*v_b(3)
      v(l,3)=akzj(l)*u_b(3)+akxj(l)*v_b(3)
      e(l,3)=0.5*r(l,3)*(u(l,3)**2+v(l,3)**2)+rgmm*p(l,3)
      grvel   =          aktj(l)
      eig11(l)= ( u_b(1) + grvel ) + osher*c(l,1)
      eig12(l)= ( u_b(2) + grvel ) + osher*c(l,2)
      eig21(l)= ( u_b(2) + grvel )
      eig22(l)= ( u_b(3) + grvel )
      eig31(l)= ( u_b(3) + grvel ) - osher*c(l,3)
      eig32(l)= ( u_b(4) + grvel ) - osher*c(l,4)
 20   continue
c*********************************************************************
c     calculate fluxes at each of the nodes for all the segments
c*********************************************************************
      do 30 m=1,4
      do 30 l=lbeg,lend
      Ucon  =fact(l)*(aktj(l)+u(l,m)*akxj(l)+v(l,m)*akzj(l))
      rUcon =r(l,m)*Ucon
      epp   =e(l,m)+p(l,m)
      pfact =fact(l)*p(l,m)
      fvs(l,1,m)=   rUcon
      fvs(l,2,m)=   rUcon*u(l,m)+akxj(l)*pfact
      fvs(l,3,m)=   rUcon*v(l,m)+akzj(l)*pfact
      fvs(l,4,m)=epp*Ucon        -aktj(l)*pfact
 30   continue
c*********************************************************************
c     calculate dfp for the first path
c*********************************************************************
      do 40 l=lbeg,lend
      az=sign(1.0,eig11(l))
      bz=sign(1.0,eig12(l))
      cz=az*bz
      dz=0.25*(cz+abs(cz))*(az+abs(az))
      dfp(l,1,1)=dz*(fvs(l,1,2)-fvs(l,1,1))
      dfp(l,2,1)=dz*(fvs(l,2,2)-fvs(l,2,1))
      dfp(l,3,1)=dz*(fvs(l,3,2)-fvs(l,3,1))
      dfp(l,4,1)=dz*(fvs(l,4,2)-fvs(l,4,1))
c*********************************************************************
c     calculate dfp for the second path
c*********************************************************************
      az=sign(1.0,eig21(l))
      dz=0.5*(az+abs(az))
      dfp(l,1,2)=dz*(fvs(l,1,3)-fvs(l,1,2))
      dfp(l,2,2)=dz*(fvs(l,2,3)-fvs(l,2,2))
      dfp(l,3,2)=dz*(fvs(l,3,3)-fvs(l,3,2))
      dfp(l,4,2)=dz*(fvs(l,4,3)-fvs(l,4,2))
c*********************************************************************
c     calculate dfp for the third path
c*********************************************************************
      az=sign(1.0,eig31(l))
```

```
      bz=sign(1.0,eig32(l))
      cz=az*bz
      dz=0.25*(cz+abs(cz))*(az+abs(az))
      dfp(l,1,3)=dz*(fvs(l,1,4)-fvs(l,1,3))
      dfp(l,2,3)=dz*(fvs(l,2,4)-fvs(l,2,3))
      dfp(l,3,3)=dz*(fvs(l,3,4)-fvs(l,3,3))
      dfp(l,4,3)=dz*(fvs(l,4,4)-fvs(l,4,3))
40    continue
c*****************************************************************
c     correction for a sonic point in first path
c*****************************************************************
      do 50 l=lbeg,lend
      az=sign(1.0,eig11(l))
      bz=sign(1.0,eig12(l))
      if((az*bz).gt.0.0) go to 50
      u_b(1)=u(l,1)*akxj(l)+v(l,1)*akzj(l)
      v_b(1)=v(l,1)*akxj(l)-u(l,1)*akzj(l)
      u_b(5)=gmbygp*(u_b(1)-osher*rexp*c(l,1))
      c(l,5)=-osher*u_b(5)
      v_b(5)=v_b(1)
c
      u(l,5)=akxj(l)*u_b(5)-akzj(l)*v_b(5)
      v(l,5)=akzj(l)*u_b(5)+akxj(l)*v_b(5)
      r(l,5)=r(l,1)*(c(l,5)/c(l,1))**rexp
      p(l,5)=r(l,5)*c(l,5)**2/gamma
      e(l,5)=0.5*r(l,5)*(u(l,5)**2+v(l,5)**2)+p(l,5)*rgmm
      m=5
      Ucon =fact(l)*(aktj(l)+u(l,m)*akxj(l)+v(l,m)*akzj(l))
      rUcon=r(l,m)*Ucon
      epp  =e(l,m)+p(l,m)
      pfact=fact(l)*p(l,m)
      fvs(l,1,m)=rUcon
      fvs(l,2,m)=rUcon*u(l,m)+akxj(l)*pfact
      fvs(l,3,m)=rUcon*v(l,m)+akzj(l)*pfact
      fvs(l,4,m)=epp*Ucon    -aktj(l)*pfact
      cz=0.5*(az+abs(az))
      dz=0.5*(bz+abs(bz))
      ez=cz-dz
      dfp(l,1,1)=dfp(l,1,1)-cz*fvs(l,1,1)+ez*fvs(l,1,5)+dz*fvs(l,1,2)
      dfp(l,2,1)=dfp(l,2,1)-cz*fvs(l,2,1)+ez*fvs(l,2,5)+dz*fvs(l,2,2)
      dfp(l,3,1)=dfp(l,3,1)-cz*fvs(l,3,1)+ez*fvs(l,3,5)+dz*fvs(l,3,2)
      dfp(l,4,1)=dfp(l,4,1)-cz*fvs(l,4,1)+ez*fvs(l,4,5)+dz*fvs(l,4,2)
50    continue
c*****************************************************************
c     correction for a sonic point in third path
c*****************************************************************
      do 60 l=lbeg,lend
      az=sign(1.0,eig31(l))
      bz=sign(1.0,eig32(l))
      if((az*bz).gt.0.0) go to 60
      u_b(4)=u(l,4)*akxj(l)+v(l,4)*akzj(l)
      v_b(4)=v(l,4)*akxj(l)-u(l,4)*akzj(l)
      u_b(6)=gmbygp*(u_b(4)+osher*rexp*c(l,4))
      c(l,6)=osher*u_b(6)
      v_b(6)=v_b(4)
c
      u(l,6)=akxj(l)*u_b(6)-akzj(l)*v_b(6)
      v(l,6)=akzj(l)*u_b(6)+akxj(l)*v_b(6)
      r(l,6)=r(l,4)*(c(l,6)/c(l,4))**rexp
      p(l,6)=r(l,6)*c(l,6)**2/gamma
      e(l,6)=0.5*r(l,6)*(u(l,6)**2+v(l,6)**2)+p(l,6)*rgmm
      m=6
      Ucon  =fact(l)*(aktj(l)+u(l,m)*akxj(l)+v(l,m)*akzj(l))
      rUcon =r(l,m)*Ucon
      epp   =e(l,m)+p(l,m)
      pfact =fact(l)*p(l,m)
      fvs(l,1,m)=   rUcon
      fvs(l,2,m)=   rUcon*u(l,m)+akxj(l)*pfact
      fvs(l,3,m)=   rUcon*v(l,m)+akzj(l)*pfact
      fvs(l,4,m)=epp*Ucon       -aktj(l)*pfact
      cz=0.5*(az+abs(az))
      dz=0.5*(bz+abs(bz))
      ez=cz-dz
      dfp(l,1,3)=dfp(l,1,3)-cz*fvs(l,1,3)+ez*fvs(l,1,6)+dz*fvs(l,1,4)
      dfp(l,2,3)=dfp(l,2,3)-cz*fvs(l,2,3)+ez*fvs(l,2,6)+dz*fvs(l,2,4)
      dfp(l,3,3)=dfp(l,3,3)-cz*fvs(l,3,3)+ez*fvs(l,3,6)+dz*fvs(l,3,4)
      dfp(l,4,3)=dfp(l,4,3)-cz*fvs(l,4,3)+ez*fvs(l,4,6)+dz*fvs(l,4,4)
60    continue
```

```
c*****************************************************************
c     calculate dfm for all the paths and fnum for the segment
c*****************************************************************
      do 70 k=1,4
      do 70 l=lbeg,lend
      dfm(l,k,1)=fvs(l,k,2)-fvs(l,k,1)-dfp(l,k,1)
      dfm(l,k,2)=fvs(l,k,3)-fvs(l,k,2)-dfp(l,k,2)
      dfm(l,k,3)=fvs(l,k,4)-fvs(l,k,3)-dfp(l,k,3)
      dfpl=dfp(l,k,1)+dfp(l,k,2)+dfp(l,k,3)
      fnum(l,k)=fvs(l,k,4)-dfpl
70    continue
c*****************************************************************
c     calculate dvq for all the paths
c*****************************************************************
      do 80 m=1,4
      do 80 l=lbeg,lend
      az=gmm/p(l,m)
      bz=az*r(l,m)
      vqs(l,1,m)=gmp-log(p(l,m)/r(l,m)**gamma)-az*e(l,m)
      vqs(l,2,m)=bz*u(l,m)
      vqs(l,3,m)=bz*v(l,m)
      vqs(l,4,m)=-bz
80    continue
      do 90 k=1,4
      do 90 l=lbeg,lend
      dvq(l,k,1)=vqs(l,k,2)-vqs(l,k,1)
      dvq(l,k,2)=vqs(l,k,3)-vqs(l,k,2)
      dvq(l,k,3)=vqs(l,k,4)-vqs(l,k,3)
90    continue
      return
      end
c-----------------------------------------------------------------
      subroutine oshvrhs
      include 'coms.f'
      common /flux  /qv(nia,2,4),fnum(nia,4),dfp(nia,4,3),dfm(nia,4,3)
     >                                      ,dvq(nia,4,3)
      nt = 1
      obyre=1.0/reynnu
      call mulam
      if ( turbl ) call eddybl
c****************************************************************
c     compute the eta direction viscous terms
c****************************************************************
      do 1000 i=1,imx(nt)
      ip=i+1
      im=i-1
      do 20   k=2,kmx(nt)
      km=k-1
      u_xi=0.0
      w_xi=0.0
      a_xi=0.0
      u0 = q(2,i,k )/q(1,i,k )
      u1 = q(2,i,km)/q(1,i,km)
      u_ze = u0 - u1
      w0 = q(3,i,k )/q(1,i,k )
      w1 = q(3,i,km)/q(1,i,km)
      w_ze = w0 - w1
      a0   =  ( q(4,i,k )/q(1,i,k ) - 0.5*( u0**2 + w0**2 ) )
      a1   =  ( q(4,i,km)/q(1,i,km) - 0.5*( u1**2 + w1**2 ) )
      a_ze =  a0-a1
c****************************************************************
c     compute the necessary metrics
c****************************************************************
      xi_x = 0.5*( xix(i,km)+xix(i,k) )
      xi_z = 0.5*( xiz(i,km)+xiz(i,k) )
      ze_x = 0.5*( zex(i,km)+zex(i,k) )
      ze_z = 0.5*( zez(i,km)+zez(i,k) )
      ajac = 0.5*( aja(i,km)+aja(i,k) )
c      ajac = 1.
c****************************************************************
c     compute the velocity derivatives w.r.t. x and z
c****************************************************************
      Ux    = ajac*( u_xi*xi_x + u_ze*ze_x )
      Wx    = ajac*( w_xi*xi_x + w_ze*ze_x )
      Ax    = ajac*( a_xi*xi_x + a_ze*ze_x )
      Uz    = ajac*( u_xi*xi_z + u_ze*ze_z )
      Wz    = ajac*( w_xi*xi_z + w_ze*ze_z )
      Az    = ajac*( a_xi*xi_z + a_ze*ze_z )
```

```
c*****************************************************************
c     compute the stress tensors
c*****************************************************************
      Vmu = 0.5*obyre*( vismu(i,km) + vismu(i,k) )
      Tmu = 0.5*obyre*( turmu(i,km) + turmu(i,k) )
      Cmu = Vmu+Tmu
      T_xx = Cmu*( 2.0*Ux-2.0*(Ux+Wz)/3.0 )
      T_zz = Cmu*( 2.0*Wz-2.0*(Ux+Wz)/3.0 )
      T_xz = Cmu*( Uz+Wx )
      Uvel = 0.5*( u0 + u1 )
      Wvel = 0.5*( w0 + w1 )
      akbycp = Vmu/prkin+Tmu/prtur
      gkbycp = gamma * akbycp
      Rx     = Uvel*T_xx + Wvel*T_xz + gkbycp*Ax
      Sz     = Uvel*T_xz + Wvel*T_zz + gkbycp*Az
c*****************************************************************
c     compute the numerical fluxes
c*****************************************************************
      fnum(k,1) = 0.0
      fnum(k,2) = ze_x*T_xx + ze_z*T_xz
      fnum(k,3) = ze_x*T_xz + ze_z*T_zz
      fnum(k,4) = ze_x*Rx   + ze_z*Sz
20    continue
      do 30 n=1,4
      do 30 k=2,kmx1(nt)
      kp0=k
      kp1=k+1
      qd(n,i,k)=qd(n,i,k)+dt(i,k)*( fnum(kp1,n)-fnum(kp0,n) )
30    continue
1000  continue
      return
      end
c-------------------------------------------------------------------
      subroutine vmatrx(jkbeg,jkend)
      parameter (nia = 213,   nka = 61)
      common /dfdq  / ap(nia,4,4), am(nia,4,4)
      common /fmet  /aktj(nia),aktnj(nia),
     >               akxj(nia),akzj(nia),ajac(nia,2),adt(nia)
      common /flux  /qv(nia,2,4),fnum(nia,4),dfp(nia,4,3),dfm(nia,4,3)
     >                                 ,dvq(nia,4,3)
      common /gamvl /gamma,            gmm,               gmp,
     >               rgamma,           rgmm,              rgmp,
     >               gmbygp
      common /tmval /timeacc, time,    dtau,dt(nia,nka),cour
      logical timeacc
      common /visdi /vmui(nka),       tmui(nka)
      common /visvl /reynnu,reynph,    prkin,             prtur
      const=1./reynnu
      rat1b3=1.0/3.0
      rat4b3=4.0/3.0
c
c     *** logic for zeta direction matrices ***
c
      do 10 jk=jkbeg,jkend
        adm    =     ajac(jk,1)
        zetax  = adm*akxj(jk)
        zetaz  = adm*akzj(jk)
        zetaxsq= zetax**2
        zetazsq= zetaz**2
c
c     *** compute the viscous parameters ***
c
        amu    = const*adt(jk)*vmui(jk)
        bmu    = const*adt(jk)*tmui(jk)
        fmu    = amu+bmu
        akbycp= amu/prkin + bmu/prtur
c
c     *** compute often used terms ***
c
        alf0 = gamma*akbycp*(zetaxsq+zetazsq)
        alf1 = fmu*(rat4b3*zetaxsq+zetazsq)
        alf3 = fmu*rat1b3*zetax*zetaz
        alf4 = fmu*(zetaxsq + zetazsq)
        alf5 = fmu*rat1b3*zetay*zetaz
        alf6 = fmu*(zetaxsq + rat4b3*zetazsq)
        rval = 0.5*(qv(jk,1,1)+qv(jk,2,1))
        obyr = 1.0/rval
        obyr1 = 1.0/qv(jk,1,1)
```

```
         obyr2 = 1.0/qv(jk,2,1)
         uval = 0.5*(qv(jk,1,2)*obyr1+qv(jk,2,2)*obyr2)
         wval = 0.5*(qv(jk,1,3)*obyr1+qv(jk,2,3)*obyr2)
         eval = 0.5*(qv(jk,1,4)+qv(jk,2,4))
         ubyro = uval*obyr
         wbyro = wval*obyr
         usqbyr = obyr*uval**2
         uwbyr = obyr*uval*wval
         wsqbyr = obyr*wval**2
         ebyrsq = eval*obyr**2
c
c     *** compute the viscous matrix ***
c
         ap(jk,1,1) = 0.0
         ap(jk,1,2) = 0.0
         ap(jk,1,3) = 0.0
         ap(jk,1,4) = 0.0
         ap(jk,2,1) = -alf1*ubyro-alf3*wbyro
         ap(jk,2,2) = alf1*obyr
         ap(jk,2,3) = alf3*obyr
         ap(jk,2,4) = 0.0
         ap(jk,3,1) = -alf3*ubyro-alf6*wbyro
         ap(jk,3,2) = alf3*obyr
         ap(jk,3,3) = alf6*obyr
         ap(jk,3,4) = 0.0
         bz = -alf1*usqbyr-alf6*wsqbyr
         cz = alf0*(usqbyr+wsqbyr-ebyrsq)
         ap(jk,4,1) = bz+cz
         ap(jk,4,2) = -ap(jk,2,1)-alf0*ubyro
         ap(jk,4,3) = -ap(jk,3,1)-alf0*wbyro
         ap(jk,4,4) = alf0*obyr
 10   continue
      return
      end
C---------------------------------------------------------------------
      subroutine smatrx(ikbeg,ikend)
      parameter (nia = 213,   nka = 61)
      common /dfdq  / ap(nia,4,4), am(nia,4,4)
      common /swvar / rsw(nia),usw(nia),vsw(nia),esw(nia)
      common /fmet  /aktj(nia),aktnj(nia),
     >               akxj(nia),akzj(nia),ajac(nia,2),adt(nia)
      common /flux  /qv(nia,2,4),fnum(nia,4),dfp(nia,4,3),dfm(nia,4,3)
     >                                      ,dvq(nia,4,3)
      common /gamvl /gamma,             gmm,                gmp,
     >               rgamma,            rgmm,               rgmp,
     >               gmbygp
      common /tmval /timeacc, time,    dtau,dt(nia,nka),cour
      logical timeacc
c
      dimension akxsq(nia),akzsq(nia),ofact(nia)
      dimension eig1(nia),eig2(nia),eig3(nia),eig4(nia),eig5(nia)
      dimension eig6(nia),eig7(nia),eig8(nia),eig9(nia),eig10(nia)
      dimension eigmd(nia,4),eigmdpl(nia,4),p(nia),c(nia),qsqby2(nia)
      eps=0.02
c
      do 10 ik=ikbeg,ikend
      qsqby2(ik)=0.5*(usw(ik)**2+vsw(ik)**2)
      p(ik)      =gmm*(esw(ik)-0.5*rsw(ik)*qsqby2(ik))
      c(ik)      =sqrt(gamma*p(ik)/rsw(ik))
      ofact(ik)  =sqrt(akxj(ik)**2+akzj(ik)**2)
      fact=1.0/ofact(ik)
      aktj(ik)=fact*aktj(ik)
      akxj(ik)=fact*akxj(ik)
      akzj(ik)=fact*akzj(ik)
      akxsq(ik)=akxj(ik)**2
      akzsq(ik)=akzj(ik)**2
      tconst=0.25*adt(ik)/sqrt(1.5)
      az=tconst*ofact(ik)
      eigmd(ik,1)=az*(aktj(ik)+usw(ik)*akxj(ik)+vsw(ik)*akzj(ik))
      eigmd(ik,2)=eigmd(ik,1)
      bz=az*c(ik)
      eigmd(ik,3)=eigmd(ik,1)+bz
      eigmd(ik,4)=eigmd(ik,1)-bz
      add=(az*eps)**2
      eigmdpl(ik,1)=sqrt(eigmd(ik,1)**2+add)
      eigmdpl(ik,2)=sqrt(eigmd(ik,2)**2+add)
      eigmdpl(ik,3)=sqrt(eigmd(ik,3)**2+add)
      eigmdpl(ik,4)=sqrt(eigmd(ik,4)**2+add)
```

```
   10 continue
      do 20 ik=ikbeg,ikend
      eigmd1=eigmd(ik,1)+eigmdpl(ik,1)
      eigmd2=eigmd(ik,2)+eigmdpl(ik,2)
      eigmd3=eigmd(ik,3)+eigmdpl(ik,3)
      eigmd4=eigmd(ik,4)+eigmdpl(ik,4)
      eig1(ik)=eigmd1+eigmd2
      eig2(ik)=eigmd3+eigmd4
      eig3(ik)=eigmd4-eigmd3
      eig4(ik)=eig2(ik)-eig1(ik)
      gbar=gmm/c(ik)**2
      eig5(ik)=gbar*eig4(ik)
      eig6(ik)=gbar*eig3(ik)*c(ik)
      eig7(ik)=eig1(ik)*akzsq(ik)+eig2(ik)*akxsq(ik)
      eig8(ik)=eig1(ik)*akxsq(ik)+eig2(ik)*akzsq(ik)
      eig9(ik)=akxj(ik)*akzj(ik)*eig4(ik)
      eig10(ik)=c(ik)*rgmm*eig3(ik)
   20 continue
      do 30 ik=ikbeg,ikend
      az=eig3(ik)/c(ik)
      ap(ik,1,1)=eig1(ik)+qsqby2(ik)*eig5(ik)
     >          +(akxj(ik)*usw(ik)+akzj(ik)*vsw(ik))*az
      ap(ik,1,2)=-usw(ik)*eig5(ik)-akxj(ik)*az
      ap(ik,1,3)=-vsw(ik)*eig5(ik)-akzj(ik)*az
      ap(ik,1,4)=eig5(ik)
      az=akxj(ik)*eig6(ik)
      ap(ik,2,1)=usw(ik)*ap(ik,1,1)-vsw(ik)*eig9(ik)
     >          -usw(ik)*eig7(ik)-qsqby2(ik)*az
      ap(ik,2,2)=usw(ik)*ap(ik,1,2)+eig7(ik)+usw(ik)*az
      ap(ik,2,3)=usw(ik)*ap(ik,1,3)+eig9(ik)+vsw(ik)*az
      ap(ik,2,4)=usw(ik)*ap(ik,1,4)-az
      az=akzj(ik)*eig6(ik)
      ap(ik,3,1)=vsw(ik)*ap(ik,1,1)-usw(ik)*eig9(ik)
     >          -vsw(ik)*eig8(ik)-qsqby2(ik)*az
      ap(ik,3,2)=vsw(ik)*ap(ik,1,2)+eig9(ik)+usw(ik)*az
      ap(ik,3,3)=vsw(ik)*ap(ik,1,3)+eig8(ik)+vsw(ik)*az
      ap(ik,3,4)=vsw(ik)*ap(ik,1,4)-az
      ap(ik,4,1)=-qsqby2(ik)*ap(ik,1,1)+usw(ik)*ap(ik,2,1)
     >           +vsw(ik)*ap(ik,3,1)+qsqby2(ik)*eig2(ik)
     >           +eig10(ik)*(usw(ik)*akxj(ik)+vsw(ik)*akzj(ik))
      ap(ik,4,2)=-qsqby2(ik)*ap(ik,1,2)+usw(ik)*ap(ik,2,2)
     >          +vsw(ik)*ap(ik,3,2)-usw(ik)*eig2(ik)-eig10(ik)*akxj(ik)
      ap(ik,4,3)=-qsqby2(ik)*ap(ik,1,3)+usw(ik)*ap(ik,2,3)
     >           +vsw(ik)*ap(ik,3,3)-vsw(ik)*eig2(ik)-eig10(ik)*akzj(ik)
      ap(ik,4,4)=-qsqby2(ik)*ap(ik,1,4)+usw(ik)*ap(ik,2,4)
     >          +vsw(ik)*ap(ik,3,4)+eig2(ik)
   30 continue
      do 40 ik=ikbeg,ikend
      eigmd1=eigmd(ik,1)-eigmdpl(ik,1)
      eigmd2=eigmd(ik,2)-eigmdpl(ik,2)
      eigmd3=eigmd(ik,3)-eigmdpl(ik,3)
      eigmd4=eigmd(ik,4)-eigmdpl(ik,4)
      eig1(ik)=eigmd1+eigmd2
      eig2(ik)=eigmd3+eigmd4
      eig3(ik)=eigmd4-eigmd3
      eig4(ik)=eig2(ik)-eig1(ik)
      gbar=gmm/c(ik)**2
      eig5(ik)=gbar*eig4(ik)
      eig6(ik)=gbar*eig3(ik)*c(ik)
      eig7(ik)=eig1(ik)*akzsq(ik)+eig2(ik)*akxsq(ik)
      eig8(ik)=eig1(ik)*akxsq(ik)+eig2(ik)*akzsq(ik)
      eig9(ik)=akxj(ik)*akzj(ik)*eig4(ik)
      eig10(ik)=c(ik)*rgmm*eig3(ik)
   40 continue
      do 50 ik=ikbeg,ikend
      az=eig3(ik)/c(ik)
      am(ik,1,1)=eig1(ik)+qsqby2(ik)*eig5(ik)
     >          +(akxj(ik)*usw(ik)+akzj(ik)*vsw(ik))*az
      am(ik,1,2)=-usw(ik)*eig5(ik)-akxj(ik)*az
      am(ik,1,3)=-vsw(ik)*eig5(ik)-akzj(ik)*az
      am(ik,1,4)=eig5(ik)
      az=akxj(ik)*eig6(ik)
      am(ik,2,1)=usw(ik)*am(ik,1,1)-vsw(ik)*eig9(ik)
     >          -usw(ik)*eig7(ik)-qsqby2(ik)*az
      am(ik,2,2)=usw(ik)*am(ik,1,2)+eig7(ik)+usw(ik)*az
      am(ik,2,3)=usw(ik)*am(ik,1,3)+eig9(ik)+vsw(ik)*az
      am(ik,2,4)=usw(ik)*am(ik,1,4)-az
      az=akzj(ik)*eig6(ik)
```

```fortran
      am(ik,3,1)=vsw(ik)*am(ik,1,1)-usw(ik)*eig9(ik)
     >          -vsw(ik)*eig8(ik)-qsqby2(ik)*az
      am(ik,3,2)=vsw(ik)*am(ik,1,2)+eig9(ik)+usw(ik)*az
      am(ik,3,3)=vsw(ik)*am(ik,1,3)+eig8(ik)+vsw(ik)*az
      am(ik,3,4)=vsw(ik)*am(ik,1,4)-az
      am(ik,4,1)=-qsqby2(ik)*am(ik,1,1)+usw(ik)*am(ik,2,1)
     >           +vsw(ik)*am(ik,3,1)+qsqby2(ik)*eig2(ik)
     >           +eig10(ik)*(usw(ik)*akxj(ik)+vsw(ik)*akzj(ik))
      am(ik,4,2)=-qsqby2(ik)*am(ik,1,2)+usw(ik)*am(ik,2,2)
     >           +vsw(ik)*am(ik,3,2)-usw(ik)*eig2(ik)-eig10(ik)*akxj(ik)
      am(ik,4,3)=-qsqby2(ik)*am(ik,1,3)+usw(ik)*am(ik,2,3)
     >           +vsw(ik)*am(ik,3,3)-vsw(ik)*eig2(ik)-eig10(ik)*akzj(ik)
      am(ik,4,4)=-qsqby2(ik)*am(ik,1,4)
     >           +usw(ik)*am(ik,2,4)+vsw(ik)*am(ik,3,4)+eig2(ik)
50    continue
      return
      end
c-----------------------------------------------------------------------
      subroutine btri(lmin,lmax,itrmax)
      parameter (nikp = 213,   ninv=61)
      common /ctri  /amat(ninv,nikp,4,4), bmat(ninv,nikp,4,4),
     >               cmat(ninv,nikp,4,4), fmat(ninv,nikp,4)
      dimension      dum(nikp,4)
      lmaxm=lmax-1
c***********************************************************************
c     lu decompose the first b block and put the elements back in this
c     b block (the diagonals contain the reciprocals of the
c     diagonals of the lower triangular matrix)
c***********************************************************************
      l=1
      do 10 i=1,itrmax
      bmat(i,1,1,1)=1.0/bmat(i,1,1,1)
      bmat(i,1,1,2)=bmat(i,1,1,1)*bmat(i,1,1,2)
      bmat(i,1,1,3)=bmat(i,1,1,1)*bmat(i,1,1,3)
      bmat(i,1,1,4)=bmat(i,1,1,1)*bmat(i,1,1,4)
      bmat(i,1,2,1)=bmat(i,1,2,1)
      bmat(i,1,2,2)=1.0/(bmat(i,1,2,2)-bmat(i,1,2,1)*bmat(i,1,1,2))
      bmat(i,1,2,3)=bmat(i,1,2,2)*(bmat(i,1,2,3)-bmat(i,1,2,1)*
     >             bmat(i,1,1,3))
      bmat(i,1,2,4)=bmat(i,1,2,2)*(bmat(i,1,2,4)-bmat(i,1,2,1)*
     >             bmat(i,1,1,4))
10    continue
      do 15 i=1,itrmax
      bmat(i,1,3,1)=bmat(i,1,3,1)
      bmat(i,1,3,2)=bmat(i,1,3,2)-bmat(i,1,3,1)*bmat(i,1,1,2)
      bmat(i,1,3,3)=1.0/(bmat(i,1,3,3)-bmat(i,1,3,1)*bmat(i,1,1,3)-
     >                  bmat(i,1,3,2)*bmat(i,1,2,3))
      bmat(i,1,3,4)=bmat(i,1,3,3)*(bmat(i,1,3,4)-bmat(i,1,3,1)*
     >             bmat(i,1,1,4)-bmat(i,1,3,2)*bmat(i,1,2,4))
      bmat(i,1,4,1)=bmat(i,1,4,1)
      bmat(i,1,4,2)=bmat(i,1,4,2)-bmat(i,1,4,1)*bmat(i,1,1,2)
      bmat(i,1,4,3)=bmat(i,1,4,3)-bmat(i,1,4,1)*bmat(i,1,1,3)-
     >             bmat(i,1,4,2)*bmat(i,1,2,3)
      bmat(i,1,4,4)=1.0/(bmat(i,1,4,4)-bmat(i,1,4,1)*bmat(i,1,1,4)-
     >                  bmat(i,1,4,2)*bmat(i,1,2,4) -
     >                  bmat(i,1,4,3)*bmat(i,1,3,4))
15    continue
c***********************************************************************
c     unitize the first b block
c***********************************************************************
      do 20 i=1,itrmax
      fmat(i,1,1)=bmat(i,1,1,1)*fmat(i,1,1)
      fmat(i,1,2)=bmat(i,1,2,2)*(fmat(i,1,2)-bmat(i,1,2,1)*fmat(i,1,1))
      fmat(i,1,3)=bmat(i,1,3,3)*(fmat(i,1,3)-bmat(i,1,3,1)*fmat(i,1,1)-
     >           bmat(i,1,3,2)*fmat(i,1,2))
      fmat(i,1,4)=bmat(i,1,4,4)*(fmat(i,1,4)-bmat(i,1,4,1)*fmat(i,1,1)-
     >           bmat(i,1,4,2)*fmat(i,1,2)-bmat(i,1,4,3)*fmat(i,1,3))
      fmat(i,1,3)=fmat(i,1,3)-bmat(i,1,3,4)*fmat(i,1,4)
      fmat(i,1,2)=fmat(i,1,2)-bmat(i,1,2,3)*fmat(i,1,3)-bmat(i,1,2,4)*
     >           fmat(i,1,4)
      fmat(i,1,1)=fmat(i,1,1)-bmat(i,1,1,2)*fmat(i,1,2)-bmat(i,1,1,3)*
     >           fmat(i,1,3)-bmat(i,1,1,4)*fmat(i,1,4)
20    continue
      do 30 m=1,4
      do 30 i=1,itrmax
      cmat(i,1,1,m)=bmat(i,1,1,1)*cmat(i,1,1,m)
      cmat(i,1,2,m)=bmat(i,1,2,2)*(cmat(i,1,2,m)-bmat(i,1,2,1)*
     >             cmat(i,1,1,m))
```

```
      cmat(i,l,3,m)=bmat(i,l,3,3)*(cmat(i,l,3,m)-bmat(i,l,3,1)*
     >            cmat(i,l,1,m)-bmat(i,l,3,2)*cmat(i,l,2,m))
      cmat(i,l,4,m)=bmat(i,l,4,4)*(cmat(i,l,4,m)-bmat(i,l,4,1)*
     >            cmat(i,l,1,m)-bmat(i,l,4,2)*cmat(i,l,2,m)-
     >            bmat(i,l,4,3)*cmat(i,l,3,m))
      cmat(i,l,3,m)=cmat(i,l,3,m)-bmat(i,l,3,4)*cmat(i,l,4,m)
      cmat(i,l,2,m)=cmat(i,l,2,m)-bmat(i,l,2,3)*cmat(i,l,3,m)-
     >            bmat(i,l,2,4)*cmat(i,l,4,m)
      cmat(i,l,1,m)=cmat(i,l,1,m)-bmat(i,l,1,2)*cmat(i,l,2,m)-
     >            bmat(i,l,1,3)*cmat(i,l,3,m)-
     >            bmat(i,l,1,4)*cmat(i,l,4,m)
30    continue
c*********************************************************************
c     upper triangularize the block tridiagonal matrix
c*********************************************************************
      do 40 l=2,lmax
      lm=l-1
c*********************************************************************
c     add -a(l)*f(l-1) to f(l) and -a(l)*c(l-1) to b(l)
c*********************************************************************
      do 50 k=1,4
      do 50 i=1,itrmax
      dum(i,k)=fmat(i,lm,k)
50    continue
      do 60 k=1,4
      do 60 i=1,itrmax
      fmat(i,l,k)=fmat(i,l,k)-
     >          amat(i,l,k,1)*dum(i,1)-amat(i,l,k,2)*dum(i,2)-
     >          amat(i,l,k,3)*dum(i,3)-amat(i,l,k,4)*dum(i,4)
60    continue
      do 70 k=1,4
      do 70 m=1,4
      do 70 i=1,itrmax
      bmat(i,l,k,m)=bmat(i,l,k,m)-amat(i,l,k,1)*cmat(i,lm,1,m)-
     >            amat(i,l,k,2)*cmat(i,lm,2,m)-
     >            amat(i,l,k,3)*cmat(i,lm,3,m)-
     >            amat(i,l,k,4)*cmat(i,lm,4,m)
70    continue
c*********************************************************************
c     lu decompose the b(l) block and put the elements back in this
c     b block (the diagonals contain the reciprocals of the
c     diagonals of the lower triangular matrix)
c*********************************************************************
      do 80 i=1,itrmax
      bmat(i,l,1,1)=1.0/bmat(i,l,1,1)
      bmat(i,l,1,2)=bmat(i,l,1,1)*bmat(i,l,1,2)
      bmat(i,l,1,3)=bmat(i,l,1,1)*bmat(i,l,1,3)
      bmat(i,l,1,4)=bmat(i,l,1,1)*bmat(i,l,1,4)
      bmat(i,l,2,1)=bmat(i,l,2,1)
      bmat(i,l,2,2)=1.0/(bmat(i,l,2,2)-bmat(i,l,2,1)*bmat(i,l,1,2))
      bmat(i,l,2,3)=bmat(i,l,2,2)*(bmat(i,l,2,3)-bmat(i,l,2,1)*
     >            bmat(i,l,1,3))
      bmat(i,l,2,4)=bmat(i,l,2,2)*(bmat(i,l,2,4)-bmat(i,l,2,1)*
     >b           mat(i,l,1,4))
80    continue
      do 85 i=1,itrmax
      bmat(i,l,3,1)=bmat(i,l,3,1)
      bmat(i,l,3,2)=bmat(i,l,3,2)-bmat(i,l,3,1)*bmat(i,l,1,2)
      bmat(i,l,3,3)=1.0/(bmat(i,l,3,3)-bmat(i,l,3,1)*bmat(i,l,1,3)-
     >                 bmat(i,l,3,2)*bmat(i,l,2,3))
      bmat(i,l,3,4)=bmat(i,l,3,3)*(bmat(i,l,3,4)-bmat(i,l,3,1)*
     >            bmat(i,l,1,4)-bmat(i,l,3,2)*bmat(i,l,2,4))
      bmat(i,l,4,1)=bmat(i,l,4,1)
      bmat(i,l,4,2)=bmat(i,l,4,2)-bmat(i,l,4,1)*bmat(i,l,1,2)
      bmat(i,l,4,3)=bmat(i,l,4,3)-bmat(i,l,4,1)*bmat(i,l,1,3)-
     >            bmat(i,l,4,2)*bmat(i,l,2,3)
      bmat(i,l,4,4)=1.0/(bmat(i,l,4,4)-bmat(i,l,4,1)*bmat(i,l,1,4)-
     >                 bmat(i,l,4,2)*bmat(i,l,2,4)-
     >                 bmat(i,l,4,3)*bmat(i,l,3,4))
85    continue
c*********************************************************************
c     unitize the b(l) block
c*********************************************************************
      do 90 i=1,itrmax
      fmat(i,l,1)=bmat(i,l,1,1)*fmat(i,l,1)
      fmat(i,l,2)=bmat(i,l,2,2)*(fmat(i,l,2)-bmat(i,l,2,1)*fmat(i,l,1))
      fmat(i,l,3)=bmat(i,l,3,3)*(fmat(i,l,3)-bmat(i,l,3,1)*fmat(i,l,1)-
     >          bmat(i,l,3,2)*fmat(i,l,2))
```

```
      fmat(i,l,4)=bmat(i,l,4,4)*(fmat(i,l,4)-bmat(i,l,4,1)*fmat(i,l,1)-
     >            bmat(i,l,4,2)*fmat(i,l,2)-bmat(i,l,4,3)*fmat(i,l,3))
      fmat(i,l,3)=fmat(i,l,3)-bmat(i,l,3,4)*fmat(i,l,4)
      fmat(i,l,2)=fmat(i,l,2)-bmat(i,l,2,3)*fmat(i,l,3)-bmat(i,l,2,4)*
     >            fmat(i,l,4)
      fmat(i,l,1)=fmat(i,l,1)-bmat(i,l,1,2)*fmat(i,l,2)-bmat(i,l,1,3)*
     >            fmat(i,l,3)-bmat(i,l,1,4)*fmat(i,l,4)
 90   continue
      do 100 m=1,4
      do 100 i=1,itrmax
      cmat(i,l,1,m)=bmat(i,l,1,1)*cmat(i,l,1,m)
      cmat(i,l,2,m)=bmat(i,l,2,2)*(cmat(i,l,2,m)-bmat(i,l,2,1)*
     >              cmat(i,l,1,m))
      cmat(i,l,3,m)=bmat(i,l,3,3)*(cmat(i,l,3,m)-bmat(i,l,3,1)*
     >              cmat(i,l,1,m)-bmat(i,l,3,2)*cmat(i,l,2,m))
      cmat(i,l,4,m)=bmat(i,l,4,4)*(cmat(i,l,4,m)-bmat(i,l,4,1)*
     >              cmat(i,l,1,m)-bmat(i,l,4,2)*cmat(i,l,2,m) -
     >              bmat(i,l,4,3)*cmat(i,l,3,m))
      cmat(i,l,3,m)=cmat(i,l,3,m)-bmat(i,l,3,4)*cmat(i,l,4,m)
      cmat(i,l,2,m)=cmat(i,l,2,m)-bmat(i,l,2,3)*cmat(i,l,3,m)-
     >                            bmat(i,l,2,4)*cmat(i,l,4,m)
      cmat(i,l,1,m)=cmat(i,l,1,m)-bmat(i,l,1,2)*cmat(i,l,2,m)-
     >                            bmat(i,l,1,3)*cmat(i,l,3,m)-
     >                            bmat(i,l,1,4)*cmat(i,l,4,m)
 100  continue
 40   continue
c*********************************************************************
c     perform the back substitution
c*********************************************************************
      do 110 l=lmaxm,1,-1
      lp=l+1
      do 120 m=1,4
      do 120 i=1,itrmax
      dum(i,m)=fmat(i,lp,m)
 120  continue
      do 110 m=1,4
      do 110 i=1,itrmax
      fmat(i,l,m)=fmat(i,l,m)-
     >            cmat(i,l,m,1)*dum(i,1)-cmat(i,l,m,2)*dum(i,2)-
     >            cmat(i,l,m,3)*dum(i,3)-cmat(i,l,m,4)*dum(i,4)
 110  continue
      return
      end
c--------------------------------------------------------------------
      subroutine mulam
      include 'coms.f'
      nt=1
      cinfsq=gamma*pinf/rinf

      do 10 i=1,imx(nt)
          do 10 k=1,kmx(nt)
            rsqqsq = q(2,i,k)**2+q(3,i,k)**2
            pval   = gmm*(q(4,i,k)-0.5*rsqqsq/q(1,i,k))
            cvalsq = gamma*pval/q(1,i,k)
            tval   = tinf*cvalsq/cinfsq
            az     = (tinf+198.6)/(tval+198.6)
            vismu(i,k) = az*(tval/tinf)**1.5
            turmu(i,k) = 0.
 10   continue
      return
      end
c--------------------------------------------------------------------
      subroutine eddybl
      include 'coms.f'

      dimension turmui(nka),turmuo(nka),fval(nka),snor(nka),vort(nka)
      dimension u_xi(nka),u_ze(nka), w_xi(nka),w_ze(nka), qtot(nka)
c     constants for the turbulence model
      aplus=26.0
      ccp=1.6
      ckleb=0.3
      cwk=0.25
      smallk=0.4
      capk=0.0168
      cmutm=14.0
      imax = imx(1)
      itel = iwks(1)
      iteu = iwke(1)
```

```
      kmax = kmx(1)
c     calculate the eddy viscosity
c..eddy is scaled down in wake
      do 10 i=itel,iteu
c     do 10 i=2,imax-1
c     calculate the magnitude of the vorticity and total velocity
      u_ze(1)=-1.5*(q(2,i,1)/q(1,i,1))+2.0*(q(2,i,2)/q(1,i,2))
     >        -0.5*(q(2,i,3)/q(1,i,3))
      w_ze(1)=-1.5*(q(3,i,1)/q(1,i,1))+2.0*(q(3,i,2)/q(1,i,2))
     >        -0.5*(q(3,i,3)/q(1,i,3))
      do 20 k=2,kmax-1
      km=k-1
      kp=k+1
      u_ze(k)=0.5*(q(2,i,kp)/q(1,i,kp)-q(2,i,km)/q(1,i,km))
      w_ze(k)=0.5*(q(3,i,kp)/q(1,i,kp)-q(3,i,km)/q(1,i,km))
 20   continue
      if(i .eq. 1) then
        do 40 k=1,kmax
        ip=i+1
        u_xi(k)=( q(2,ip,k)/q(1,ip,k)-q(2,i,k)/q(1,i,k) )
        w_xi(k)=( q(3,ip,k)/q(1,ip,k)-q(3,i,k)/q(1,i,k) )
 40     continue
      else
        do 60 k=1,kmax-1
        ip=i+1
        im=i-1
        u_xi(k)=0.5*( q(2,ip,k)/q(1,ip,k)-q(2,im,k)/q(1,im,k) )
        w_xi(k)=0.5*( q(3,ip,k)/q(1,ip,k)-q(3,im,k)/q(1,im,k) )
 60     continue
      endif
      do 70 k=1,kmax-1
      dudz=( u_xi(k)*xiz(i,k)+u_ze(k)*zez(i,k) )*aja(i,k)
      dwdx=( w_xi(k)*xix(i,k)+w_ze(k)*zex(i,k) )*aja(i,k)
      vort(k)=abs(dudz-dwdx)
      qtot(k)=sqrt(q(2,i,k)**2+q(3,i,k)**2)/q(1,i,k)
 70   continue
c
c     calculate the distance normal to the body
c
      snor(1)=0.0
      do 80 k=2,kmax
      km=k-1
      az=x(i,k)-x(i,km)
      bz=z(i,k)-z(i,km)
      snor(k)=snor(km)+sqrt(az**2+bz**2)
 80   continue
c     calculate the exponent for the exponential term
      k=1
c..by vorticity...
c     yac = aja(i,k)
c     ux=( u_xi(k)*xix(i,k)+u_ze(k)*zex(i,k) ) * yac
c     wx=( w_xi(k)*xix(i,k)+w_ze(k)*zex(i,k) ) * yac
c     uz=( u_xi(k)*xiz(i,k)+u_ze(k)*zez(i,k) ) * yac
c     wz=( w_xi(k)*xiz(i,k)+w_ze(k)*zez(i,k) ) * yac
c     fmu=vismu(i,k)
c     tauxx=fmu*(2.0*ux-2.0*(ux+wz)/3.0)
c     tauxz=fmu*(uz+wx)
c     tauzz=fmu*(2.0*wz-2.0*(ux+wz)/3.0)
c     ze_x = zex(i,k)
c     ze_z = zez(i,k)
c     fact= 1.0/sqrt( ze_x**2+ze_z**2)
c     ak1 = fact*ze_z
c     ak2 =-fact*ze_x
c     tauwal=abs((tauxx-tauzz)*ak1*ak2+tauxz*(ak2**2-ak1**2))
c     expnnt=sqrt(q(1,i,k)*tauwal)/(vismu(i,k)*aplus)
c     expnnt=expnnt*sqrt(reynnu)
      EXPNNT = SQRT( REYNNU*q(1,I,K)*VORT(K) ) / (VISMU(I,K)*APLUS)
c     calculate the eddy viscosity fot the iiner region
c     mt_inner = rho * (l**2) * vort
      do 90 k=1,kmax-1
      alen=smallk*snor(k)*(1.0-exp(-expnnt*snor(k)))
      turmui(k)=reynnu*q(1,i,k)*vort(k)*alen**2
 90   continue
c     calculate the eddy viscosity for the outer region
      do k=1,kmax-1
      fval(k)=snor(k)*vort(k)*(1.0-exp(-expnnt*snor(k)))
      enddo
      fmax=0.0
```

```
      do 110 k=3,kmax-1
      if(fmax.le.fval(k) .or. fmax.lt.fval(k+1)) go to 115
      ypl=expnnt*aplus*snor(k)
      if(ypl.gt.30) go to 120
 115  continue
      fmax=fval(k)
      ks=k
 110  continue
 120  continue
      ksm=ks-1
      ksp=ks+1
      az=(fval(ks)-fval(ksm))/(snor(ks)-snor(ksm))
      bz=(fval(ksp)-fval(ks))/(snor(ksp)-snor(ks))
      aval=(bz-az)/(snor(ksp)-snor(ksm))
      bval=az-aval*(snor(ks)+snor(ksm))
      aval=aval+1.0e-08*sign(1.0,aval)
      snormx=-0.5*bval/aval
      snormx=max(snormx,snor(ksm))
      snormx=min(snormx,snor(ksp))
      klft=ksm
      if(snormx.gt.snor(ks)) klft=ks
      krgt=klft+1
      frc=(snormx-snor(klft))/(snor(krgt)-snor(klft))
      fmax=fval(klft)+frc*(fval(krgt)-fval(klft))
      qmax=-100000.0
      qmin=100000.0
      do 130 k=1,kmax-1
      qmax=max(qmax,qtot(k))
      qmin=min(qmin,qtot(k))
 130  continue
      qdif=qmax-qmin
      az=snormx*fmax
      bz=cwk*snormx*qdif*qdif/(fmax+1.0e-08*sign(1.0,fmax))
c     fwake=min(az,bz)
      if( i .gt. itel .and. i .lt. iteu ) then
        fwake=az
      else
        fwake=bz
      endif
      const=capk*ccp*fwake*reynnu
      do k=1,kmax-1
        fkleb=1.0/(1.0+5.5*(ckleb*snor(k)/snormx)**6)
        turmuo(k)=const*fkleb*q(1,i,k)
      enddo
c..choose from the inner and outer eddy viscosity values
      ainner=1.0
      do k=1,kmax-1
      if(turmui(k).gt.turmuo(k)) ainner=0.0
      turmu(i,k)=ainner*turmui(k)+(1.0-ainner)*turmuo(k)
      enddo
 10   continue
c..eddy is scaled down in wake
      do i = 2,itel
      iu  = imax - i + 1
      fac = 1./(1.+ (x(i,1)-x(itel,1))**3)
      do k = 1,kmax-1
      turmu(i,k)  = turmu(itel+1,k)*fac
      turmu(iu,k) = turmu(iteu-1,k)*fac
      enddo
      enddo
c     if(itr .eq. niter)  then
c       open(unit=30,file='turmu.d',form='formatted')
c       ip1 = 71
c       ip2 = 100
c       ip3 = 101
c       ip4 = 102
c       ip5 = 115
c       write(30,'(f5.2,5e14.6)') ( float(k),
c    >   turmu(ip1,k), turmu(ip2,k), turmu(ip3,k),
c    >   turmu(ip4,k), turmu(ip5,k),
c    >     k=2,kmx(1))
c       endif
      return
      end
C-------------------------------------------------------------------
      subroutine metric
      include 'coms.f'
      dimension ajamax(nia),ajamin(nia)
```

```
      data eps /1.e-26/
c*******************************************************************
      nt = 1
      ng = 1
      do 100 i=1,imx(nt)
        im1 = i - 1
        ip1 = i + 1
      do 100 k = 1,kmx(nt)
        if( i .eq. 1 ) then
          xxi = x(2,k) - x(1,k)
          zxi = z(2,k) - z(1,k)
        elseif( i .eq. imx(nt) ) then
          xxi = x(imx(nt),k) - x(imx1(nt),k)
          zxi = z(imx(nt),k) - z(imx1(nt),k)
        else
          xxi = 0.5 * ( x(ip1,k) - x(im1,k) )
          zxi = 0.5 * ( z(ip1,k) - z(im1,k) )
        endif
        if( k .eq. 1 ) then
          xze = 2.*x(i,2) - 1.5*x(i,1) - 0.5*x(i,3)
          zze = 2.*z(i,2) - 1.5*z(i,1) - 0.5*z(i,3)
        elseif( k .eq. kmx(nt) ) then
          xze = 1.5*x(i,kmx(nt)) - 2.*x(i,kmx1(nt)) + 0.5*x(i,kmx2(nt))
          zze = 1.5*z(i,kmx(nt)) - 2.*z(i,kmx1(nt)) + 0.5*z(i,kmx2(nt))
        else
          km1 = k - 1
          kp1 = k + 1
          xze = 0.5 * ( x(i,kp1) - x(i,km1) )
          zze = 0.5 * ( z(i,kp1) - z(i,km1) )
        endif
        xix(i,k) = zze
        xiz(i,k) =-xze
        zex(i,k) =-zxi
        zez(i,k) = xxi
c       xix(i,k) = bjac * zze
c       xiz(i,k) =-bjac * xze
c       zex(i,k) =-bjac * zxi
c       zez(i,k) = bjac * xxi
        xdot =  omega * z(i,k)
        zdot = -omega * x(i,k)
        xit(i,k) = -xdot*xix(i,k) - zdot*xiz(i,k)
        zet(i,k) = -xdot*zex(i,k) - zdot*zez(i,k)
        yacob = ( xxi*zze - xze*zxi )
        if ( yacob .eq. 0. ) then
          print *, 'zero jac at ', i,k
           yacob = eps
        endif
        aja(i,k) = 1.0 / yacob
 100  continue
      if( oscil .or. ramp ) return
c*******************************************************************
c         compute max and min values of jacobian and check for
c         negative values
c*******************************************************************
      ajmax      = -1.0e35
      ajmin      =  1.0e35
      do 63 k = 1,kmx(ng)
      do 63 i = 1,imx(ng)
      ajmax      = max( ajmax,aja(i,k) )
      ajmin      = min( ajmin,aja(i,k) )
   63 continue
      write(6,602) ajmax,ajmin
c..write negative jacobians and stop
      if( ajmin.lt.0.0 ) then
          do 64 k = 1,kmx(ng)
          do 64 i = 1,imx(ng)
          if( aja(i,k).lt.0.0 ) then
              write(6,603) aja(i,k), i, k
              stop
          end if
   64 continue
      end if
  602 format( ' The range of the jacobian is: ',
     >         ' jmax = ',e10.3,5x,'jmin =  ',e10.3,/    )
  603 format( ' ',10x,'negative jacobian = ',e10.3,1x,'at i,k =',
     >                 2i5 )
      return
```

```
      end
C-----------------------------------------------------------------------
      subroutine eigen
      include 'coms.f'
      almax=0.0
c..compute the maximum eigenvalue
      nt = 1
      ng = 1
      do 10 i = 2, imxl(nt)
        ip   = i + 1
        im   = i - 1
c..evaluate the derivatives of x and z for the line ***
          do 20 k = 2, kmxl(nt)
            kp     = k + 1
            km     = k - 1
            xta = 0.0
            zta = 0.0
            xps = 0.5*(x(ip,k) - x(im,k))
            zps = 0.5*(z(ip,k) - z(im,k))
            xze = 0.5*(x(i,kp) - x(i,km))
            zze = 0.5*(z(i,kp) - z(i,km))
c..compute the maximum eigenvalue ***
            bjac       = abs(1.0/( xps*zze - xze*zps ) )
ccc            bjac = aja(i,k)
ccc            psixj =  bjac * zze
ccc            psizj = -bjac * xze
ccc            zetxj = -bjac * zps
ccc            zetzj =  bjac * xps
            psixj = xps
            psizj = xze
            zetxj = zps
            zetzj = zze
            obyr       = 1.0/q(1,i,k)
            rqsq       = obyr*(q(2,i,k)**2 + q(3,i,k)**2 )
            pval       = gmm *(q(4,i,k) - 0.5*rqsq)
            cval       = sqrt(gamma*pval*obyr)
            uvel       = obyr*q(2,i,k) - xta
            wvel       = obyr*q(3,i,k) - zta
            ucon       = abs(uvel*psixj + wvel*psizj)
            bz         = cval*sqrt(psixj**2 + psizj**2)
            alpsi      = bjac*(ucon + bz)
            vcon       = abs(uvel*zetxj + wvel*zetzj)
            bz         = cval*sqrt(zetxj**2 + zetzj**2)
            alzet      = bjac*(vcon + bz)
            almaxn     = sqrt(alpsi**2 + alzet**2)
            almax      = max(almaxn, almax)
20        continue
10    continue
        dtau=cour/almax
      if( timeacc ) then
        do  k=1,kmx(1)
        do  i=1,imx(1)
        dt(i,k) =  dtau
       enddo
       enddo
      else
c..evaluate variable dtau scaling based on Jacobian..
        do  k=1,kmx(1)
        do  i=1,imx(1)
        sqjac    = sqrt( aja(i,k) )
        dt(i,k) = ( 1.0 + dtau*sqjac )/( 1.0+sqjac )
        enddo
       enddo
      endif
      print *, 'L_max  = ', almax
      write (6,61) dtau
61    format ( ' dtau   = ', f12.8)
      return
      end
C-----------------------------------------------------------------------
      subroutine loads
 Comment !!!
 **   THIS SUBROUTINE IS INCORRECT !!!
 **   Must use non-rotated grid x(i,k), z(i,k)
 **   See wrcl.f
      include 'coms.f'
      dimension cp(nia) , cf(nia), txzs(nia), yplus(nia)
      itel = iwks(1)
```

```
      iteu = iwke(1)
 c..compute pressure loads
      cpc    =  1. / ( 0.5*rinf*uinf**2)
      do 100 i = itel, iteu
      p        = gmm*( q(4,i,1)
     >                - .5*(q(2,i,1)**2 + q(3,i,1)**2)/q(1,i,1) )
      cp(i)    = - (p - pinf) * cpc
100   continue
      cn  = 0.0
      ch  = 0.0
      cm  = 0.0
      do 25 i=itel,iteu-1
          dx =  x(i+1,1) - x(i,1)
          dz =  z(i+1,1) - z(i,1)
          avcp = 0.5*( cp(i+1)+cp(i) )
          cn = cn + avcp*dx
          ch = ch - avcp*dz
c.. cm about 25% chord
          cm = cm - avcp * ( dz*z(i,1) + dx*(x(i,1)-.25) )
   25   continue
      cl = cn*cos(alfa) - ch*sin(alfa)
      cd = cn*sin(alfa) + ch*cos(alfa)
c..compute viscous loads
      do 10 i =itel, iteu
       u_xi = 0.0
       u_ze = q(2,i,2)/q(1,i,2) - q(2,i,1)/q(1,i,1)
       w_xi = 0.0
       w_ze = q(3,i,2)/q(1,i,2) - q(3,i,1)/q(1,i,1)
       xi_x = xix(i,1)*aja(i,1)
       ze_x = zex(i,1)*aja(i,1)
       xi_z = xiz(i,1)*aja(i,1)
       ze_z = zex(i,1)*aja(i,1)
       u_x = u_xi * xi_x + u_ze * ze_x
       w_x = w_xi * xi_x + w_ze * ze_x
       u_z = u_xi * xi_z + u_ze * ze_z
       w_z = w_xi * xi_z + w_ze * ze_z
       viscl       = 0.5*( vismu(i,1) + vismu(i,2) )
       visct       = 0.5*( turmu(i,1) + turmu(i,2) )
       viscto        = (viscl + visct) / reynnu
       txzs(i)   = ( (viscto*(u_z + w_x)) / (0.5 * amach**2) )
c..skin friction
       sn = sqrt( (x(i,2)-x(i,3))**2 +  (z(i,2)-z(i,3))**2 )
       rho = q(i,1,1)
       yplus(i) = sqrt( abs(txzs(i)) * rho ) * sn  / viscto
       dx = (x(i+1,1) - x(i,1))
       dz = (z(i+1,1) - z(i,1))
       cf(i) = -txzs(i)*(dz/abs(dz)) * 1000
   10 continue
      cnv = 0.
      chv = 0.
      cmv = 0.
      do 20 i = itel, iteu-1
       dx = (x(i+1,1) - x(i,1))
       dz = (z(i+1,1) - z(i,1))
       avtxzs = 0.5*( txzs(i+1)+txzs(i) )
       cnv = cnv + avtxzs*dz
       chv = chv + avtxzs*dx
       cmv = cmv + avtxzs * (dx * z(i,1) - dz * x(i,1) )
20    continue
      clv = cnv*cos(alfa) - chv*sin(alfa)
      cdv = cnv*sin(alfa) + chv*cos(alfa)
      write(9,101) iter, alfad, time, amach, reynph,
     >             itel, iteu, cl,cd,cm, clv,cdv,cmv,
     >             (cp(i), i=itel,iteu),(cf(i), i=itel,iteu)
  101 format(i10,4e12.4, 2i5/ 6e12.4 / (6e12.3) )
      return
      end
C-----------------------------------------------------------------
      subroutine grmove(dalfa)
      include 'coms.f'
      if( dalfa .eq. 0.) return
      ca = cos( dalfa )
      sa =-sin( dalfa )
      do 10 i=1,imx(1)
      do 10 k=1,kmx(1)
      xold = x(i,k)
      zold = z(i,k)
      x(i,k) = xold * ca - zold * sa
```

```
      z(i,k) = zold * ca + xold * sa
 10   continue
      call metric
      return
      end
C-----------------------------------------------------------------------
      subroutine qio(io)
      include 'coms.f'
      IF ( IO .eq. 0) THEN
      open(unit=32,file='ends.d',form='unformatted')
      write (32) imx(1), kmx(1), ksi
      write (32) amach,alfad,reynph,time,iter
      write (32) ((( q(l,i,k), i=1,imx(1) ), k=1,kmx(1) ), l=1,4)
      close(32)
      ELSEIF (IO .eq. 10) THEN
      write (8) imx(1), kmx(1), ksi
      write (8) amach,alfad,reynph,time,iter
      write (8) ((( q(l,i,k), i=1,imx(1) ), k=1,kmx(1) ), l=1,4)
      ELSEIF (IO .eq. 1) THEN
      open(unit=31,file='strs.d',form='unformatted',status='old')
      read  (31) imx(1), kmx(1), ksi
      read  (31) amachr,alfad,reynphr,time,iter
      read  (31) ((( q(l,i,k), i=1,imx(1) ), k=1,kmx(1) ), l=1,4)
      close(31)
      kso = kmx(1)
      ELSEIF (IO .eq. 2) THEN
      open(unit=31,file='strs.d',form='formatted',status='old')
      read  (31,*) imx(1), kmx(1), ksi
      read  (31,*) amachr,alfad,reynphr,time,iter
      read  (31,*) ((( q(l,i,k), i=1,imx(1) ), k=1,kmx(1) ), l=1,4)
      close(31)
      kso = kmx(1)
      ENDIF
      return
      end
```

APPENDIX D

A. MICHEL'S EMPIRICAL CORRELATION METHOD

A problem was discovered with the subroutine 'output' in BL2D.F that calculates the transition location using the Michel's empirical correlation. The computed transition location was found to be computer and input transition point dependent. This apparent computational error did not affect any other computational results.

The Michel empirical correlation is based on incompressible, constant property flow over a flat plate, and is presented in Equation D.1 with chord (c) assumed to be one (Cebeci and Bradshaw [Ref. 5]).

$$R_{\theta_{tr}} = 1.174 \left[1 + \frac{22,400}{R_{e_{x_{tr}}}}\right] R_{e_{x_{tr}}}^{.46} \qquad \text{(D.1)}$$

$$R_{e_{x_{tr}}} = \frac{U_e x_{tr}}{\nu} = \frac{U_e}{U_\infty} x_{tr} R_e$$

$$R_e = \frac{\rho U_\infty c}{\mu} = \frac{U_\infty c}{\nu} = \frac{U_\infty}{\nu} \qquad \text{(D.2)}$$

$\frac{U_e}{U_\infty}$ = *Normalized Velocity on* i^{th} *Panel*

The functional relationship between momentum thickness and transition Reynolds number is presented in Equation D.3. Solving Equations D.1 and D.3 simultaneously yields the results shown in Figure D.1.

$$R_{\theta_{tr}} = 0.664 \sqrt{R_{e_{x_{tr}}}} \qquad \text{(D.3)}$$

An alternate approach for the solution of $R_{\theta_{tr}}$ is shown in Equation D.4 and D.5 with $\rho/\rho_e=1$ for incompressible flow.

$$R_{\theta_{tr}} = \frac{U_e\,\theta_{tr}}{\nu} = \frac{U_e}{U_\infty}\,\theta_{tr}\,R_e \qquad \text{(D.4)}$$

$$\begin{aligned} \theta_{tr} &= \int_0^\delta \left[\frac{\rho u}{\rho_e u_e}\left(1-\frac{u}{u_e}\right)\right]dy \\ \delta^* &= \int_0^\delta \left(1-\frac{\rho u}{\rho_e u_e}\right)dy \\ \delta^* &\approx 0.3\ \delta \end{aligned} \qquad \text{(D.5)}$$

The BL2D.F program uses Equations D.1 and D.4 to find the transition location. Each panel on a surface is checked by computing the transition Reynolds number, momentum thickness, and Equations D.1 and D.4. The panel where Equation D.1 is

approximately equal to Equation D.4 is identified as the transition location (surface distance from the input stagnation point). Both the Indigo and Stardent computers compute Equation D.1 exactly the same as can be observed in Figures D.2 and D.3. However, the summing routines used to calculate Equations D.4 and D.5 are computed differently depending on the machine used due to precision differences, thus producing different transition locations with the same input parameters (Figure D.3).

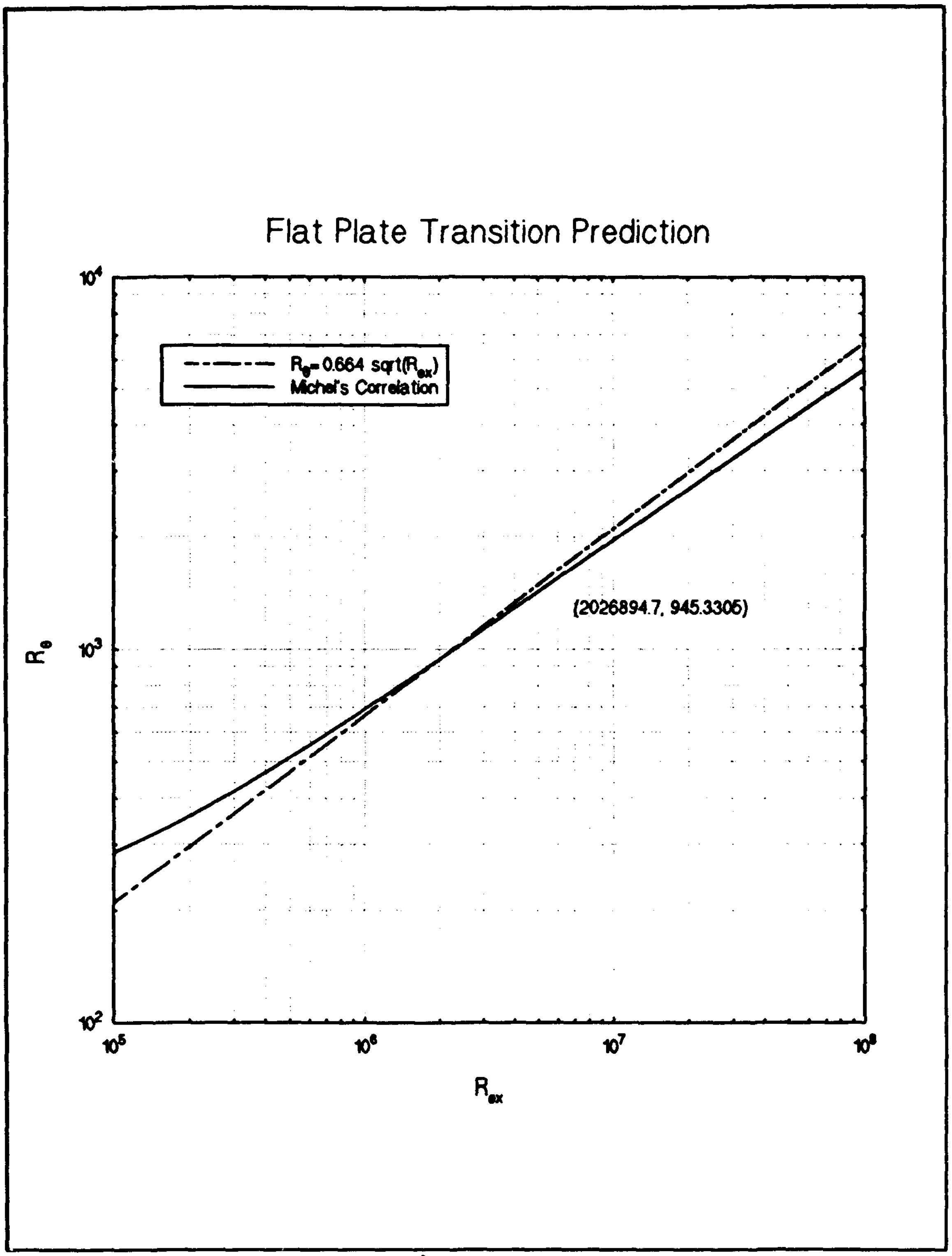

Flat Plate Transition Prediction
$R_\theta = 0.664\ \mathrm{sqrt}(R_{ex})$
Michel's Correlation
(2026894.7, 945.3305)
R_θ
R_{ex}
10^4
10^3
10^2
10^5
10^6
10^7
10^8

Figure D.1

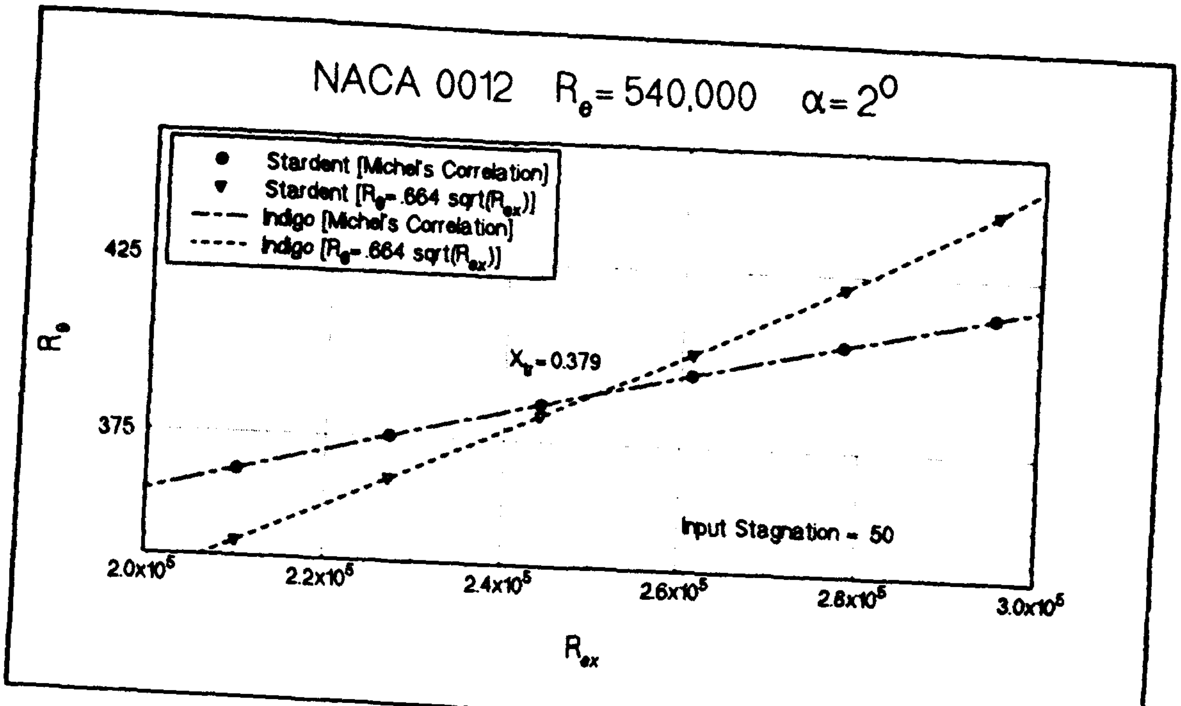


Figure D.2

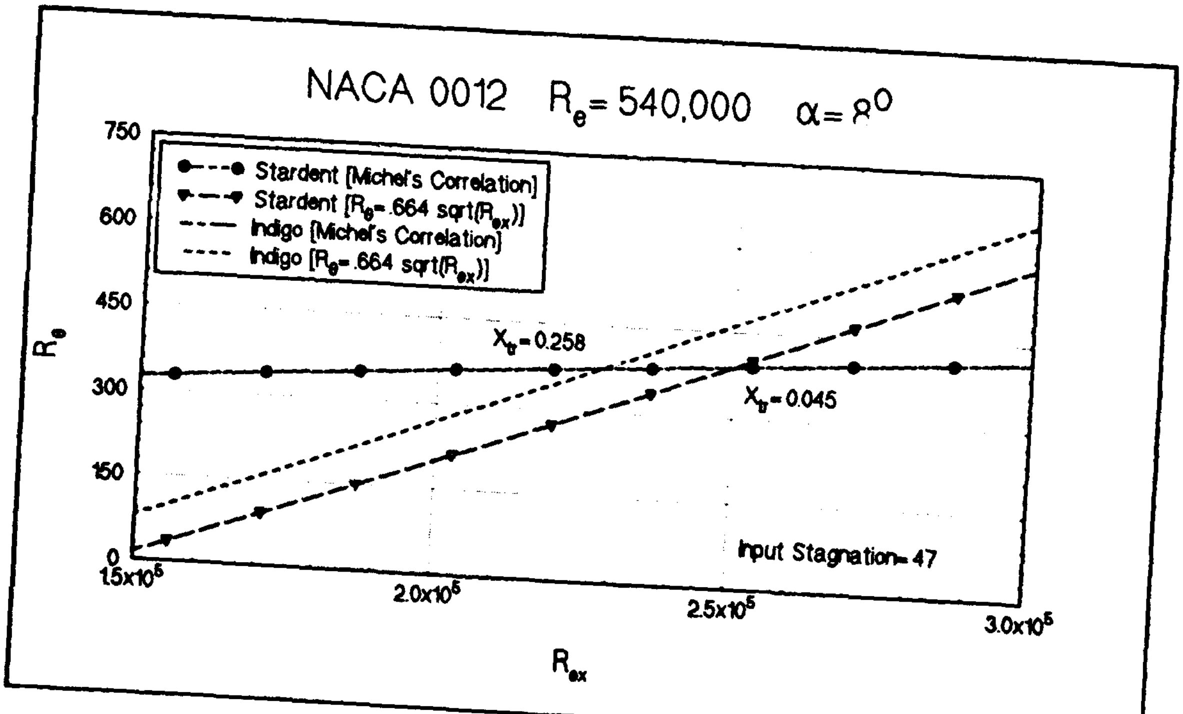


Figure D.3

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 052 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Dr. M. F. Plazer Dept. of Aeronautics and Astronautics, Code AA/Pl Naval Postgraduate School Monterey, California 93943-5002	7
4.	Dr. J. A. Ekaterinaris M. S. 260-1 NASA Ames Research Center Moffett Field, California 94035	1
5.	CDR T. A. Johnston, USN PMA 273 Room 852 JP1 1421 Jefferson Davis Highway Arlington, Va 22243-1273	3